Becky Stewart

À l'aventure avec ARDUINO

Dès 10 ans

Découvre Arduino et l'électronique grâce à 9 aventures trépidantes !

EYROLLES

Copyright © 2015 Eyrolles.

À l'aventure avec ARDUINO

Pars à la découverte d'Arduino !

Amusant et facile à lire, ce livre te fera découvrir l'électronique et surtout le formidable potentiel d'Arduino, un petit microcontrôleur programmable qui permet de réaliser plein de projets ludiques. Avec des composants simples, du carton et de la colle, tu seras guidé dans la mise en œuvre de montages de difficulté croissante, allant d'un panneau d'affichage de LED à un jeu de labyrinthe à bille. Combinant avec intelligence théorie et pratique, cet ouvrage fera de toi un pro d'Arduino et des circuits électroniques. Dans notre société toujours plus numérique, cela te donnera un super avantage pour la suite de ton parcours scolaire.

Dans ce livre, tu fabriqueras :

- un carillon à vent électronique
- un mini coffre-fort à combinaison
- une boule de cristal qui s'illumine par magie
- un labyrinthe à bille qui mémorise ton score
- un affichage lumineux sur la manche d'un vêtement
- et bien d'autres projets !

À propos de l'auteur

Ingénieur et professeur, Becky Stewart est la cofondatrice de Codasign, un organisme de formation qui propose des ateliers sur Arduino et les technologies numériques à l'attention du jeune public.

À qui s'adresse cet ouvrage ?
Aux 10-15 ans, parents et enseignants.

Sur www.editions-eyrolles.com/go/arduino
Télécharge toutes les ressources (code source, vidéos) du livre.

Copyright © 2015 Eyrolles.

À l'aventure avec ARDUINO

Copyright © 2015 Eyrolles.

M. O'Hanlon et D. Wale. – **Apprendre à coder en Python grâce à Minecraft.** *À paraître.*
N°14292, 2016, 350 pages.

Dans la collection « Pour les kids »

D. Weinachter. – **Cahier d'activités Python pour les kids.** *Dès 10 ans.*
N°14332, 2016, 64 pages.

J. R. Briggs. – **Python pour les kids.** *Dès 10 ans.*
N°14088, 2015, 332 pages.

F. Pain. – **Cahier d'activités Scratch pour les kids.** *Dès 8 ans.*
N°14218, 2015, 64 pages.

The LEAD Project. – **Scratch pour les kids.** *Dès 8 ans.*
N°14117, 2015, 160 pages.

Dans la collection « Serial Makers »

E. Bartmann. – **Le grand livre d'Arduino (2e édition).**
N°14117, 2015, 586 pages.

C. Platt. – **L'électronique en pratique 2.** *36 expériences ludiques.*
N°14179, 2015, 336 pages.

C. Platt. – **L'électronique en pratique.** *36 expériences ludiques.*
N°13507, 2013, 344 pages.

J. Boyer. – **Réparez vous-même vos appareils électroniques.**
N°13936, 2014, 384 pages.

M. Richardson et S. Wallace. – **À la découverte du Raspberry Pi.**
N°13747, 2013, 176 pages.

A. Banks, MacUser et iFixit. – **Réparez vous-même votre Apple.** *Une seconde vie pour votre iPhone, MacBook, iMac…*
N°14251, 2015, 146 pages.

F. Botton. – **Les drones de loisir.**
N°14219, 2015, 224 pages.

R. Jobard. – **Les drones.** *La nouvelle révolution.*
N°13976, 2014, 190 pages.

B. Pettis, A. Kaziunas France et J. Shergill. – **Imprimer en 3D avec la MakerBot.**
N°13748, 2013, 226 pages.

M. Berchon. – **L'impression 3D (2e édition).**
N°13946, 2014, 232 pages.

C. Bosqué, O. Noor et L. Ricard. – **FabLabs, etc.** *Les nouveaux lieux de fabrication numérique.*
N°13938, 2015, 216 pages.

Copyright © 2015 Eyrolles.

BECKY STEWART

À l'aventure avec ARDUINO

EYROLLES

Copyright © 2015 Eyrolles.

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

© 2015 John Wiley and Sons, Ltd.

All Rights Reserved. Authorised translation from the English language edition published by John Wiley & Sons Limited. Responsibility for the accuracy of the translation rests solely with Groupe Eyrolles and is not the responsibility of John Wiley & Sons Limited. No part of this book may be reproduced in any form without the written permission of the original copyright holder John Wiley & Sons Limited.

Traduction autorisée de l'ouvrage en langue anglaise intitulé *Adventures in Arduino* par Becky Stewart (ISBN 978-1-118-94847-7), publié par John Wiley & Sons.

Traduit et adapté de l'anglais par Jean Boyer.

Copyright © 2015 Eyrolles.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.
© Groupe Eyrolles, 2015, ISBN : 978-2-212-14314-0

Aux étudiants à qui l'on a toujours dit
qu'ils devaient choisir entre les arts et la science.

Remerciements

Je remercie tout d'abord Alexandra Deschamps-Sonsino, sans qui je n'aurais jamais écrit ce livre. J'exprime également ma sincère gratitude à Alex pour son travail avec Tinker qui a donné le premier élan à la communauté Arduino de Londres.

Rien de tout cela n'aurait été possible sans le soutien des personnels d'éducation de Codasign. Ainsi, grâce à Melissa Coleman et Pollie Barden, j'ai beaucoup appris sur la manière d'enseigner Arduino, et j'apprends constamment, avec Adam Stark, à mieux former les enfants à la programmation informatique. Je n'oublierai pas non plus de remercier Emilie Giles, car Codasign aurait stagné sans vous : merci pour tout ce que vous avez fait. Je remercie aussi tout particulièrement Liat Wassershtrom, pour ses retours et son expertise.

Je voudrais également remercier les équipes éditoriales et artistiques qui m'ont aidée à mettre en forme cet ouvrage. Il a été grandement amélioré grâce à vos conseils.

Pour terminer, merci à ma famille qui m'a soutenue dans tout ce que j'ai entrepris. Merci à mes parents, qui ont mis à ma disposition un endroit calme pour écrire. Merci enfin à Ben, qui a gentiment accepté d'ajuster les dates de notre mariage et de nos vacances en fonction de mes travaux d'écriture.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Table des matières

Introduction 1

Qu'est-ce qu'Arduino ? 1
Que vas-tu apprendre dans ce livre ? 2
De quoi auras-tu besoin ? 2
Les outils nécessaires 11
Les logiciels indispensables 15
Autre matériel utile 15
Que dois-tu connaître avant de lire ce livre ? 16
Comment ce livre est organisé ? 16
Présentation du texte 18
Le site web d'accompagnement 19
Pour obtenir de l'aide 20
Un mot du traducteur 20

Aventure 1

Installe et teste ton Arduino 21

Ce dont tu as besoin 22
Téléchargement et installation du logiciel Arduino sur ton ordinateur 22
Installation du logiciel Arduino sur un Mac 24
Installation du logiciel Arduino sur un PC (Windows) 25
Installation du logiciel Arduino sur un ordinateur Linux 27
À la découverte de l'IDE Arduino 28
Vérification de l'installation avec le test du clignotement 29
Téléchargement du code de clignotement 31
Recherche des problèmes courants 34
Réalisation d'un circuit avec une LED 41
Ce dont tu as besoin 41
Compréhension des schémas des circuits 42
Utilisation d'une plaque de montage 44
Construction de ton premier circuit 45
Plus d'aventures en Arduino 48

Aventure 2

Utilise des capteurs 51

Ce dont tu as besoin 51
Ajout de plusieurs LED 52

Copyright © 2015 Eyrolles.

Envoi de messages à ton ordinateur 60
Lecture de données en provenance d'un potentiomètre 65
Comment prendre des décisions dans ton code ? 69
Construction d'un panneau d'affichage de messages 70
Ce dont tu as besoin 71
Compréhension du circuit 72
Réalisation du prototype sur ta plaque de montage 72
Écriture du programme 73
Construction de ton panneau d'affichage 79
Plus d'aventures avec Arduino 85

Aventure 3

Utilise des servomoteurs 87

Ce dont tu as besoin 87
Description des différents types de moteurs 89
Contrôle d'un servomoteur avec Arduino 90
Répétition d'actions 93
Entrée numérique avec un bouton-poussoir 95
Construction de ton coffre-fort à combinaison 99
Ce dont tu as besoin 100
Comprendre le circuit 102
Le prototype sur plaque de montage 102
Écriture du programme 104
Réalisation du coffre-fort 114
Soudage des fils 116
Insertion de l'électronique 118
Plus d'aventures avec Arduino 119

Aventure 4

Utilise des registres à décalage 121

Ce dont tu as besoin 122
Organisation de ton code 123
Utilisation des fonctions 123
Utilisation des boucles for 128
Comment obtenir plus de sorties avec un registre à décalage ? 130
Comment fonctionne un registre à décalage ? 132
Réalisation des connexions de ton registre à décalage 134
Ajout de LED 135
Écriture du programme 137

Copyright © 2015 Eyrolles.

Ajout d'autres registres à décalage 143
Réalisation de l'affichage lumineux de ton nom 145
Ce dont tu as besoin. 147
Comprendre le circuit 148
Réalisation du prototype sur plaque de montage 149
Écriture du programme 149
Fabrication de tes lettres lumineuses 155
Soudage des fils. 155
Insertion du circuit électronique 157
Plus d'aventures avec les registres à décalage 157

Aventure 5
Diffuse des sons. 159
Ce dont tu as besoin 159
Création d'une liste 160
Dévoile tes intentions 162
Utilise une boucle et un tableau. 164
Mets cela en pratique. 164
Production de bruit. 168
Câblage de ton circuit 170
Écriture du programme 170
Réalisation d'un carillon à vent électronique 176
Ce dont tu as besoin. 178
Comprendre le circuit 179
Réalisation du prototype. 179
Écriture du programme 181
Fabrication de ton carillon à vent 182
Plus d'aventures sonores 188

Aventure 6
Ajoute des bibliothèques 191
Ce dont tu as besoin 191
Les sorties analogiques. 192
Contrôle de la brillance d'une LED 195
Mélange de lumières colorées. 198
Les capteurs capacitifs 208
Ajout d'une bibliothèque. 208
Réalisation du câblage de ton circuit 210
Écriture du programme 211

Copyright © 2015 Eyrolles.

Fabrication de ta boule de cristal 215
Comprendre le circuit 216
Réalisation du prototype sur plaque de montage 217
Écriture du programme 218
Construction de ta boule de cristal 222
Plus d'aventures avec les bibliothèques 225

Aventure 7
Utilise l'Arduino Leonardo 227

Ce dont tu as besoin 227
Découverte de l'Arduino Leonardo 228
Première connexion de ta carte Arduino Leonardo 228
Un comportement identique à celui d'un clavier 231
Détection de la lumière 235
Réalisation de ton circuit 236
Écriture du programme 238
Réalisation d'une manette de jeu 240
Ce dont tu as besoin 241
Construction de ton circuit 241
Écriture du programme 243
Fabrication du boîtier de ta manette de jeu 253
Assemble le tout 254
Plus d'aventures avec l'Arduino Leonardo 255

Aventure 8
Utilise l'Arduino LilyPad USB 257

Ce dont tu as besoin 258
Découverte de l'Arduino LilyPad USB 258
Un clignotant utilisant une carte Arduino LilyPad 260
Prototypage des circuits souples 262
Utiliser les tableaux astucieusement 264
Passage de données entre fonctions 270
Réalisation d'un vêtement à persistance de vision 273
Ce dont tu as besoin 274
Comprendre le circuit 274
Écriture du programme 277
Fabrication de ton vêtement à vision persistante 280
Plus d'aventures avec l'Arduino LilyPad 283

Copyright © 2015 Eyrolles.

Aventure 9

La grande aventure : un labyrinthe à bille. .285

Ce dont tu as besoin 286
Première partie : le décompte des points 287
Détection des vibrations avec des capteurs piézoélectriques 287
Définir un seuil pour le marquage des points 289
Ajouter des effets sonores 292
Mémoriser le score 293
Deuxième partie : création du labyrinthe 295
Dessiner le labyrinthe 296
Créer les règles du jeu 297
Réaliser le prototype du circuit 298
Troisième partie : écriture du programme 300
Début de partie 300
Fin de partie 301
Début d'une nouvelle partie 304
Mémoriser le score maximal 306
Ajouter les sons 307
Quatrième partie : fabrication du labyrinthe à bille 315
Fabriquer ton labyrinthe 315
Assembler les composants piézoélectriques 317
Assembler les LED et le bouton-poussoir 318
La touche finale 319
Encore plus d'aventures : va plus loin avec Arduino 320

Annexe A

Pour aller plus loin 321

Plus de cartes, d'interfaces, de capteurs et d'actionneurs 321
Les cartes interfaces (shields) 322
Les capteurs et les actionneurs 322
Tutoriels sur le Web 322
Le site officiel d'Arduino 322
Les sites de fabricants 323
Les blogs 323
Les vidéos 323
Livres 324
Sur Arduino 324
Sur l'électronique de base 324

Copyright © 2015 Eyrolles.

Sur les circuits souples et les projets vestimentaires. 325
Autres sujets spécialisés 325

Annexe B
Où se procurer outils et composants 327
Les kits de démarrage 327
Les magasins spécialisés. 328
Les boutiques en ligne 328
Les boutiques en ligne expédiant depuis l'Europe. 328
Les boutiques en ligne expédiant depuis les USA ou le Canada 329
Deux autres sites incontournables 329

Glossaire. 331

Index 337

Copyright © 2015 Eyrolles.

Introduction

ES-TU UN aventurier ? As-tu envie de créer des projets qui t'amèneront à développer des nouvelles compétences et à maîtriser des outils inconnus ? Veux-tu découvrir comment utiliser la technologie pour concrétiser tes idées ? Es-tu curieux de savoir comment combiner programmation informatique et circuits électroniques avec des ciseaux et du papier, et même avec une aiguille et du fil ? Si la réponse est sans hésitation « oui », alors ce livre est fait pour toi !

Qu'est-ce qu'Arduino ?

Arduino est un outil permettant de construire des dispositifs qui peuvent interagir avec l'environnement qui les entoure. Tu peux t'en servir pour y relier des capteurs détectant du son, de la lumière ou des vibrations, qu'il utilisera alors pour allumer une lumière, changer sa couleur, mettre en route un moteur, et bien d'autres choses. Arduino est un système magique, qui se situe au cœur de toutes ces actions. Il collecte des informations à partir de ses capteurs, évaluant ainsi le monde réel qui l'entoure. Il prend ensuite des décisions basées sur les données recueillies et provoque en retour des actions, sous forme de sons, de lumière, ou encore de mouvements.

Arduino se présente généralement sous la forme d'une carte électronique bleue, qui a à peu près la taille de ta main. Cette carte comporte des inscriptions en blanc qui permettent de repérer ses différents éléments. Tous les composants et les circuits de la carte sont visibles et accessibles. Il existe plusieurs types de cartes Arduino, et elles ne sont pas toutes bleues. Mais tu en apprendras davantage dans la section « De quoi auras-tu besoin ? » page 2, ainsi que dans les aventures 7 et 8.

Arduino est un microcontrôleur, autrement dit un ordinateur très simple. Il ne peut pas faire beaucoup de choses en même temps, mais ce qu'on lui dit de faire, il le fait très bien. Sans le savoir, tu utilises déjà beaucoup de microcontrôleurs dans la vie de tous les jours, car ils sont présents dans beaucoup d'appareils, comme les fours à micro-ondes et les machines à laver.

Il existe de nombreux types de microcontrôleurs mais ce qui est particulier avec Arduino, c'est qu'il est conçu pour être utilisé par les débutants. Donc, si tu es novice en programmation ou en électronique, tout va bien. Mais attention, ne sous-estime pas Arduino pour autant : il peut aussi s'adapter à de gros projets !

Copyright © 2015 Eyrolles.

Que vas-tu apprendre dans ce livre ?

Lorsque tu seras arrivé à la fin des aventures de ce livre, tu auras appris à installer l'environnement de programmation d'Arduino sur ton ordinateur, à l'utiliser pour écrire un programme et à le télécharger dans ton Arduino. Tu auras aussi découvert comment travailler avec trois types de cartes Arduino : la Uno, la Leonardo et la LilyPad USB.

Tu auras également appris les concepts de base de la programmation informatique, que tu pourras ensuite mettre en œuvre sur d'autres systèmes qu'Arduino. En effet, le langage d'Arduino est basé sur le langage C/C++. Cela signifie qu'en apprenant à programmer Arduino, tu sauras également comment s'effectue la programmation sur un ordinateur portable ou sur un Raspberry Pi.

En plus de la programmation, tu te seras familiarisé avec l'électronique et plus particulièrement avec ses différents circuits. Tu auras appris comment utiliser des capteurs pour détecter les signaux provenant du monde qui nous entoure, tels que la lumière ou les mouvements. Et tu sauras comment générer des actions en retour, par exemple produire un son ou allumer une lampe.

À la fin du livre, tu auras ainsi acquis une large connaissance de ce qui peut être réalisé avec Arduino. Tu seras alors prêt à concevoir et réaliser tes propres projets !

De quoi auras-tu besoin ?

Cette section décrit tous les éléments dont tu auras besoin pour réaliser les projets de ce livre, sachant que de nombreux projets utilisent les mêmes composants.

Il est de plus en plus facile de se procurer des cartes Arduino (on en trouve même à la librairie Eyrolles !). Les boutiques spécialisées en électronique, comme Selectronic à Paris et à Lille, en ont toujours en stock. Ces magasins commercialisent également les autres composants dont tu auras besoin pour les projets de ce livre. Si tu ne peux pas te rendre dans une de ces boutiques, va sur Internet car de nombreux sites web proposent ces produits, ils sont répertoriés dans l'annexe B.

Le composant le plus important dont tu auras besoin est, bien entendu, une carte Arduino. Il existe différentes cartes Arduino, mais la Uno est la plus répandue et c'est celle que tu utiliseras le plus dans ce livre. Tu auras également besoin d'une carte Arduino Leonardo pour l'aventure 7 et d'une carte Arduino LilyPad USB pour l'aventure 8. Ces différentes cartes sont présentées sur la figure I-1.

Tu auras en plus besoin d'un câble USB pour relier la carte Arduino à ton ordinateur. Pour l'Arduino Uno, il s'agira d'un câble standard ; pour la Leonardo et la LilyPad USB, un câble micro USB sera nécessaire. Les deux sont visibles sur la figure I-2.

Copyright © 2015 Eyrolles.

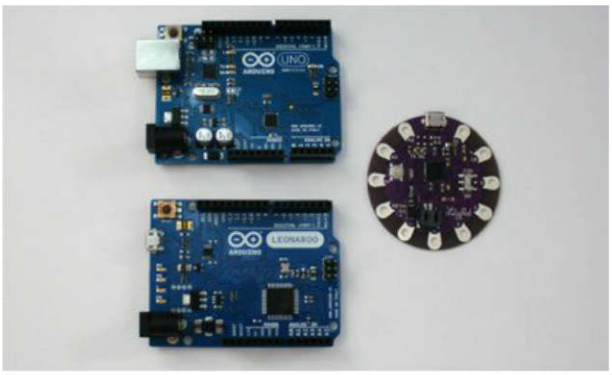


FIGURE I-1 L'Arduino Uno (en haut à gauche), l'Arduino Leonardo (en bas à gauche) et l'Arduino LilyPad USB (à droite).

FIGURE I-2 Un câble USB standard et un câble micro USB.

Tu vas aussi utiliser des plaques de montage pour relier facilement des composants électroniques sans avoir à les souder. Ces plaques existent dans différentes couleurs et dimensions. Les plus grandes sont utiles pour les projets les plus compliqués, qui nécessitent de nombreux composants ; les plus petites seront à utiliser pour les projets que tu souhaiteras réaliser dans un espace limité. Deux plaques de tailles différentes, réalisées avec des plastiques différents, sont montrées sur la figure I-3.

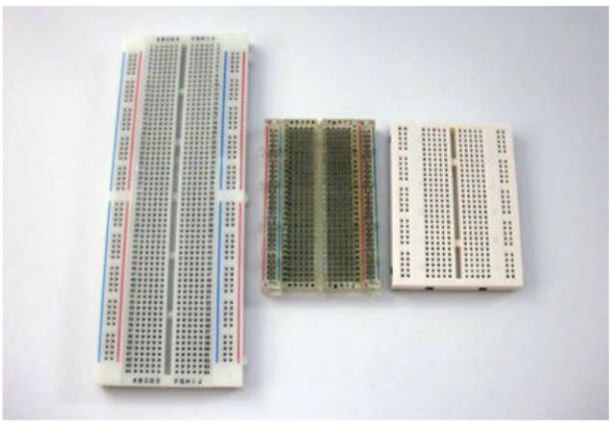


FIGURE I-3 Des plaques de montage de diverses dimensions et couleurs.

Copyright © 2015 Eyrolles.

L'aventure 3 est le seul projet qui utilise une plaque de montage pour sa réalisation. Les autres aventures n'en utilisent une que pour tester leur circuit. Il sera plus facile de travailler avec une plus grande plaque, mais si tu ne peux t'en procurer qu'une petite, cela conviendra très bien.

Les fils de pontage sont des fils que tu utiliseras pour réaliser les prototypes de tes circuits pour tester de nouveaux concepts. Certains sont rigides et de petite longueur, comme ceux montrés sur la figure I-4, à droite. D'autres sont plus souples avec des broches de connexion à chaque extrémité, comme ceux de gauche.

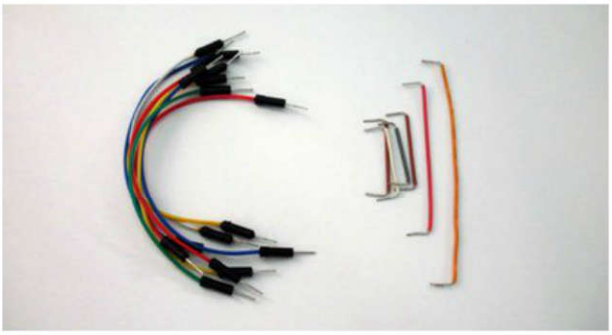


FIGURE I-4 Des fils de pontage.

Les LED sont de petites lampes spéciales (LED signifie *Light Emitting Diode*, ou diode électroluminescente) qui sont disponibles dans une large sélection de tailles et de couleurs. Pour la plupart des projets de ce livre, tu pourras utiliser n'importe quelle taille et couleur, selon ta préférence. La taille la plus commune est de 5 mm de diamètre. Cependant, une diode de taille plus importante (10 mm, par exemple) pourra également être utilisée, pour plus de plaisir. La majorité des LED sont de couleur unique, mais, dans l'aventure 6, tu utiliseras une LED possédant quatre fils au lieu de deux : cette LED peut changer de couleur. Dans l'aventure 8, tu utiliseras un composant appelé LED LilyPad, spécialement conçu pour les circuits cousus sur du tissu.

Toutes les LED utilisées dans les projets de ce livre sont présentées sur la figure I-5.

FIGURE I-5 Différents types de LED dont une LED à changement de couleur (à droite) et une LED LilyPad (en dessous).

Copyright © 2015 Eyrolles.

Les résistances sont des composants qui seront décrits dans les aventures. Elles sont disponibles avec différentes valeurs de résistance, mesurées en ohms (Ω). Tu n'auras pas besoin de beaucoup de résistances différentes pour les projets de ce livre. Les résistances étant de petits composants de faible prix, il serait bon d'en acheter des quantités plus importantes que celles requises. Tu auras besoin de résistances de 68 ou 100 Ω, 220 Ω, 10 kΩ (10 000 Ω), 1 MΩ (1 000 000 Ω) et 10 MΩ (10 000 000 Ω). La figure I-6 montre les différentes résistances dont tu auras besoin.

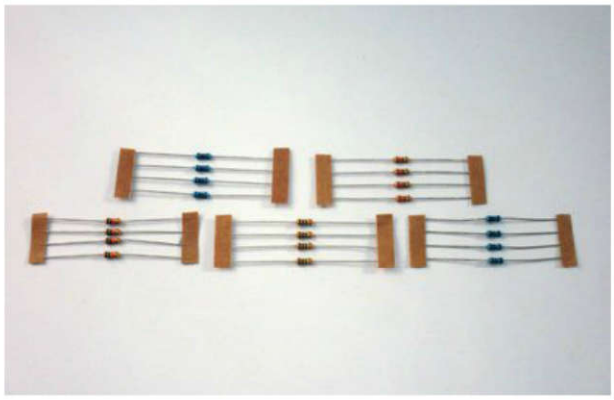


FIGURE I-6 Les résistances nécessaires aux projets de ce livre : 100 Ω (en haut à gauche), 220 Ω (en haut à droite), 10 kΩ (en bas à gauche), 1 MΩ (en bas au milieu) et 10 MΩ (en bas à droite).

Les potentiomètres sont des composants électroniques que l'on utilise par exemple pour le réglage du volume sonore dans les appareils stéréophoniques. Ils se présentent sous différentes formes et dimensions. Certains peuvent s'insérer dans une plaque de montage, comme celui en bleu sur la figure I-7. D'autres nécessitent des fils soudés qui permettront de les relier à la plaque de montage, comme celui qui est au centre de la figure. Les modèles plus gros sont plus faciles à utiliser dans un projet, ce sont des potentiomètres pour montage en façade de boîtier.

FIGURE I-7 Trois types de potentiomètres.

Copyright © 2015 Eyrolles.

Un servomoteur, comme celui de la figure I-8, est un type de moteur que tu utiliseras dans l'aventure 3.

FIGURE I-8 Un servomoteur.

Les boutons-poussoirs sont d'autres composants se présentant sous différentes formes et dimensions. Tu n'as peut-être pas remarqué cela auparavant, mais il en existe de nombreuses sortes ! Tous les projets de ce livre utilisent des boutons-poussoirs qui ferment un contact (à l'opposé des boutons-poussoirs qui en ouvrent un). C'est donc des boutons de ce type qu'il conviendra d'acheter. Tant qu'il s'agit de poussoirs fermant un contact, tu pourras en utiliser n'importe quelle sorte. Les boutons-poussoirs tactiles sont de très petits interrupteurs qui peuvent s'insérer dans les plaques de montage ; ils sont pratiques pour tester les circuits. Pour les projets de ce livre, des boutons-poussoirs prévus pour être montés sur la façade d'un boîtier sont préférables. Les deux types sont présentés sur la figure I-9.

FIGURE I-9 Un bouton-poussoir tactile (à gauche) et trois boutons-poussoirs différents prévus pour un montage en façade (à droite).

Dans l'aventure 4, tu apprendras à utiliser les registres à décalage. Ce sont de petits circuits intégrés noirs que tu peux utiliser pour contrôler un grand nombre de LED. Tu

Copyright © 2015 Eyrolles.

auras besoin d'un circuit intégré de type registre à décalage 74HC595 (tu verras ce que cela signifie plus tard). C'est ce circuit intégré que tu devras acheter, il possède 16 broches, comme celui de la figure I-10.

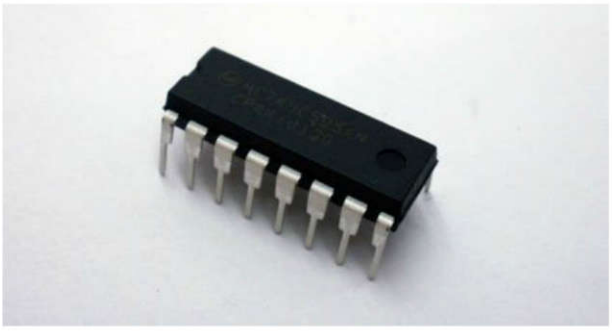


FIGURE I-10 Un registre à décalage.

Les transducteurs piézoélectriques sont des composants utilisés pour détecter des vibrations ou pour reproduire des sons, comme ceux d'un haut-parleur. Tu auras besoin d'un transducteur pour les aventures 5, 6 et 9. Ces composants se présentent parfois sous forme d'un boîtier en plastique noir, ce qui conviendra pour l'aventure 5, mais tu devras aussi disposer d'au moins cinq transducteurs sans boîtier (comme celui de la figure I-11), pour l'aventure 9.

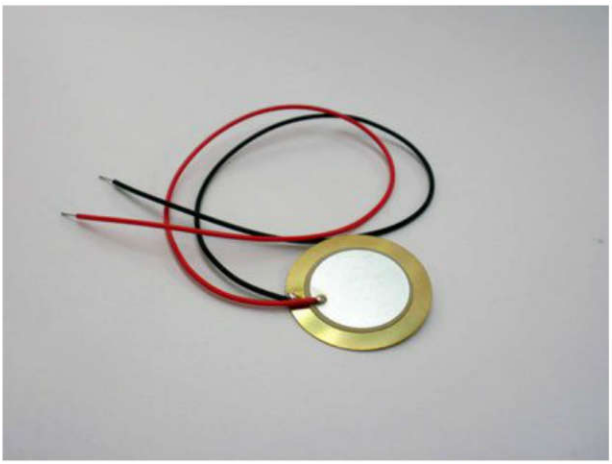


FIGURE I-11 Un transducteur piézoélectrique.

Une cellule photorésistive indique à Arduino la luminosité ambiante. Ce composant ressemble à celui de la figure I-12, d'autres peuvent être un peu plus grands.

Les barrettes de connexion de ce composant sont des petites broches métalliques de contact séparées par du plastique qui s'insèrent parfaitement dans les trous de connexion d'une carte Arduino Uno. Comme elles existent selon différents espacements, tu feras attention de bien choisir des barrettes à contacts mâles avec un

Copyright © 2015 Eyrolles.

espacement de 2,54 mm, comme ceux de la figure I-13. Tu auras besoin d'une barrette de cinq contacts pour l'aventure 5. Tu pourras aussi t'en procurer des plus longues et les découper à l'aide d'une pince coupante.

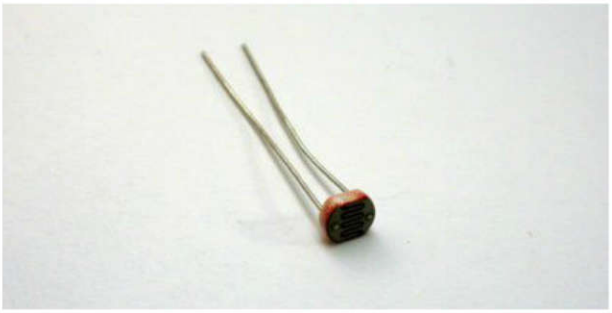


FIGURE I-12 Une cellule photorésistive.

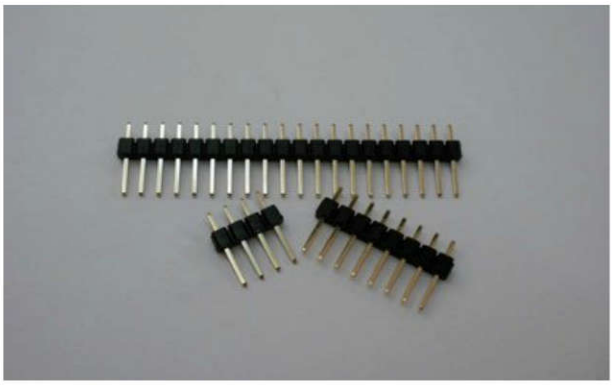


FIGURE I-13 Des barrettes à contacts mâles.

Lorsque tu penses aux circuits électriques, la première chose qui te vient à l'esprit est probablement un fil. Mais sais-tu qu'il en existe différentes sortes ? Un fil peut être constitué d'un seul brin métallique (fil rigide) ou de plusieurs brins torsadés (fil multi-brins). La figure I-14 montre ces deux types de fils.

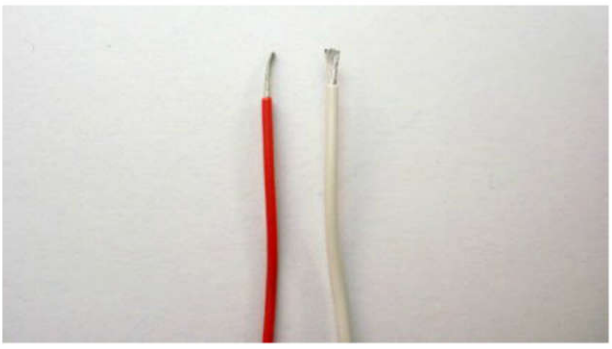


FIGURE I-14 Du fil rigide (à gauche) et du fil multibrins (à droite).

Copyright © 2015 Eyrolles.

Les fils rigides sont utiles avec des plaques de montage, mais ils sont peu souples. Les fils multibrins sont plus faciles à plier, mais tu devras étamer leurs extrémités à l'aide d'un fer à souder pour pouvoir les insérer dans une plaque de montage. Tu pourras décider, pour chaque projet, le type de fil que tu préfères : il n'y a pas de bon ou mauvais type à utiliser.

Les fils sont en général revêtus d'une enveloppe isolante ne conduisant pas l'électricité. Cette enveloppe peut être en plastique, comme pour le fil de droite sur la figure I-15, ou un vernis, comme le fil de gauche (qu'on appelle fil émaillé). Tu pourras choisir le fil qui te convient le mieux pour tes projets. Le fil émaillé conviendra bien pour le carillon de l'aventure 5, car il est très fin et permettra aux timbres du carillon de se balancer librement. Mais tu pourras aussi réaliser ce carillon en utilisant un autre type de fil très fin.

FIGURE I-15 Du fil émaillé (à gauche) et du fil plastifié (à droite).

En électronique, la soudure est comparable à une colle conductrice. Elle se présente sous forme de bobines de fil à souder de différents diamètres, comme sur la figure I-16. Les projets de ce livre n'ont pas besoin de soudure spécifique, tu n'auras donc pas à te soucier du diamètre du fil que tu achèteras : n'importe lequel conviendra. Le seul point important sera de t'assurer d'acheter de la soudure pour l'électronique. N'achète surtout pas de la soudure pour la plomberie !

FIGURE I-16 Des bobines de fil à souder.

Copyright © 2015 Eyrolles.

Après avoir réalisé tes projets, tu souhaiteras peut-être les faire fonctionner sans devoir les relier à ton ordinateur pour les alimenter en électricité. Tu pourras alors les alimenter avec une alimentation électrique ou une pile. Dans le premier cas, le plus simple sera d'utiliser un bloc d'alimentation branché sur le secteur électrique et possédant une sortie USB qui te permettra de relier le bloc secteur et ta carte Arduino avec un câble USB. Si tu souhaites utiliser une pile, le plus simple est d'en utiliser une de 9 V (volts) avec un adaptateur à fiche d'alimentation cylindrique ; sur la carte Arduino se trouve un connecteur noir, dans lequel tu peux brancher la fiche de l'adaptateur de la pile. Les deux possibilités sont montrées sur la figure I-17. Pour la carte LilyPad USB, tu pourras utiliser une batterie LiPo : tu le verras dans l'aventure 8.

FIGURE I-17 Un adaptateur de pile 9 V avec fiche cylindrique (à gauche) et un bloc d'alimentation secteur à sortie USB (à droite).

Pour réaliser les circuits de l'aventure 8, tu auras besoin de fils munis de pinces crocodiles, comme ceux de la figure I-18. Tu utiliseras à la place des fils de pontage.

FIGURE I-18 Des fils munis de pinces crocodiles.

Copyright © 2015 Eyrolles.

Dans cette aventure, tu utiliseras également de la cordelette conductrice. Il en existe différentes sortes, mais elles sont toutes de couleur argent, comme celles de la figure I-19.

FIGURE I-19 De la cordelette conductrice.

Les outils nécessaires

Tout comme tu aurais besoin d'un marteau et d'une scie pour fabriquer un objet en bois, tu auras besoin d'outils spécifiques pour réaliser tes circuits électroniques.

Tu utiliseras une plaque de montage pour tester tes circuits, mais auras parfois besoin de t'étendre au-delà de la plaque. Ce sera le cas, par exemple, quand tu voudras ajouter des fils plus longs à un composant pour qu'il puisse se loger à l'intérieur du boîtier de ton projet. Il en sera de même quand tu souhaiteras fixer des composants entre eux, afin qu'ils ne puissent pas se détacher de ton montage.

Le premier outil indispensable est un fer à souder. La soudure est comparable à de la colle pour l'électronique, mais elle nécessite une haute température pour être appliquée (imagine une colle chaude pour l'électronique). Un fer à souder est un outil qui devient très chaud (plus chaud qu'un four) afin de faire fondre le fil à souder. N'utilise un fer à souder qu'en présence d'un adulte qui pourra t'aider.

Un fer à souder peut se brancher sur une prise électrique et se tenant d'une seule main, comme celui de la figure I-20. Il peut aussi s'insérer dans un boîtier connecté au secteur, avec réglage de température. Les deux types sont adaptés. Le plus important est d'en acheter un prévu pour la soudure de circuits électroniques et non pour la plomberie ou toute autre activité.

Les fils se présentent le plus souvent avec une gaine en plastique qui est un isolant ne conduisant pas l'électricité. Tu auras parfois à retirer cette gaine aux extrémités d'un fil afin de lui permettre de s'insérer dans une plaque de montage ou pour le souder à un

Copyright © 2015 Eyrolles.

composant. Tu pourras essayer d'utiliser un couteau ou un cutter avec précaution pour retirer l'isolant plastique, mais cela risque d'être très frustrant. Il est bien préférable d'acheter l'outil adéquat pour cette opération : la pince à dénuder !

FIGURE I-20 Un fer à souder.

Les pinces à dénuder se présentent selon différentes formes et tailles, comme tu peux le voir sur la figure I-21. Tu pourras choisir celle qui te convient le mieux.

FIGURE I-21 Différents modèles de pinces à dénuder.

Les pinces coupantes, tu t'en doutes, coupent les fils. Assure-toi d'en trouver des petites, qui tiennent facilement dans ta main, car tu vas devoir travailler avec des composants et des fils de petites tailles. La figure I-22 montre le modèle de pinces coupantes que tu pourrais acheter.

Les pinces pliantes t'aideront à plier et donner forme aux fils, elles existent dans différentes formes et dimensions. Pour les projets de ce livre, tu auras besoin d'une petite pince pliante à usage général. Toutes les pinces de la figure I-23 conviendront.

Copyright © 2015 Eyrolles.

FIGURE I-22 Une pince coupante.

FIGURE I-23 Deux modèles de pince pliante.

L'outil qui suit peut paraître un peu intimidant, mais il pourra devenir ton meilleur compagnon quand tu travailleras avec des circuits électroniques : c'est le multimètre ! Il mesure plusieurs paramètres (c'est pour cela qu'on le nomme ainsi) dont les résistances et les tensions, qui sont les plus utiles pour un débutant. On en trouve des très peu coûteux comme des très chers. Quand tu en choisiras un pour ton travail, tu n'auras pas à dépenser beaucoup, en particulier si c'est ton premier multimètre. En cherchant un multimètre en magasin, tu trouveras parfois la fonction de calibre automatique nommée *autorange*. Le calibre automatique signifie que tu n'as pas à connaître la valeur approximative de la mesure effectuée avant de la réaliser : le multimètre s'y adapte automatiquement. Par contre, tu devras absolument t'assurer qu'il permet d'effectuer des tests de continuité. Un test de continuité fait retentir un buzzer quand une connexion électrique est établie entre les deux fils de mesure du multimètre. La figure I-24 montre un multimètre peu coûteux, ne possédant pas le calibre automatique mais assurant les tests de continuité.

Copyright © 2015 Eyrolles.

Les derniers outils qui te seront nécessaires ne sont pas spécifiques à l'électronique, mais ils seront essentiels pour construire les boîtiers de tes projets : des ciseaux et un couteau universel (figure I-25). Fais bien attention à ne pas te blesser en les utilisant !

FIGURE I-24 Un multimètre.

FIGURE I-25 Une paire de ciseaux et un couteau universel.

Copyright © 2015 Eyrolles.

Les logiciels indispensables

En parlant d'Arduino, il est facile de se limiter à la carte du circuit. C'est après tout la partie que tu vas physiquement placer dans tes projets. Cependant, Arduino a besoin d'un programme codifié pour pouvoir fonctionner. Tu devras d'abord écrire ce code à l'aide d'un ordinateur, puis le télécharger (dans le vocabulaire d'Arduino, on dit aussi « téléverser ») sur ta carte Arduino pour qu'il s'exécute.

La société qui fabrique les cartes Arduino conçoit également le logiciel qui t'aidera à écrire puis à téléverser ton code. Tu peux télécharger gratuitement ce logiciel depuis le site : `http://arduino.cc/en/Main/Software`

L'aventure 1 t'indiquera les étapes à suivre pour installer le logiciel sur ton ordinateur.

Les schémas des circuits ainsi que les plans de câblage des plaques de montage de ce livre ont été réalisés à l'aide d'un logiciel appelé Fritzing, qui est disponible gratuitement sur ce site : `http://fritzing.org/download`

Tu pourras également utiliser Fritzing pour concevoir tes propres projets.

Autre matériel utile

Écrire un programme et construire un circuit ne représentent que la moitié d'un projet. Ce dernier ne prendra réellement forme que lorsqu'il sera placé dans une sorte de boîtier. Que ce soit un jeu ou une lampe interactive, tant qu'il n'est constitué que d'un circuit sur une plaque de montage, il n'a pas atteint tout son potentiel.

Pour réaliser les projets de ce livre, tu devras donc employer des techniques simples, en plus de tout ce que tu auras appris en électronique. Des ciseaux, du papier et de la colle constitueront ainsi le matériel de base pour de nombreuses aventures. De plus, tes projets seront encore plus beaux et plus personnels si tu leur ajoutes une petite touche de déco. En particulier, tu auras besoin des éléments suivants :

- une petite boîte en carton, ou une boîte à chaussures ;
- du papier cartonné, du carton et du papier ;
- de la peinture pour peindre des objets ;
- de la ficelle ou du fil ;
- de la colle blanche, un bâtonnet de colle ou un pistolet à colle ;
- un pinceau ;
- un ballon (pour l'aventure 6) ;
- une bille (pour l'aventure 9).

Copyright © 2015 Eyrolles.

Que dois-tu connaître avant de lire ce livre ?

Puisque tu as commencé la lecture de ce livre, je pense que tu dois t'intéresser à la technologie ! Il n'est pas nécessaire que tu aies déjà fait de la programmation sur ordinateur ou construit des circuits (puisque c'est ce que ce livre explique !). En revanche, je suppose que tu as déjà utilisé un ordinateur auparavant.

Tu auras besoin d'un ordinateur afin de travailler avec Arduino, mais son système d'exploitation importe peu – Mac OS X, Windows ou les différentes distributions de Linux (voir `http://playground.arduino.cc/Learning/Linux` pour connaître les distributions Linux qui peuvent être utilisées). Je suppose aussi que tu sais utiliser Internet et télécharger des fichiers, et que tu sais trouver et ouvrir les applications présentes sur un ordinateur.

Tu auras peut-être besoin du mot de passe de l'administrateur de ton ordinateur pour installer les logiciels. Si tu ne le connais pas, demande l'aide de quelqu'un qui le connaît quand tu les installeras.

Comment ce livre est organisé ?

Ce livre va te guider dans la programmation de ta carte Arduino et dans la construction des circuits des neuf aventures. Chaque aventure commence par une introduction aux nouvelles connaissances dont tu auras besoin pour réaliser le projet proposé à la fin. Ces compétences peuvent correspondre par exemple à la découverte d'un nouveau composant, comme un moteur, ou encore à l'écriture intelligente d'un programme.

L'aventure la plus importante pour commencer est bien entendu l'aventure 1, car elle t'aidera à installer le logiciel nécessaire au téléchargement de ton code dans la carte Arduino. Si tu ne commences pas par cette étape, le reste des aventures n'aura que peu d'intérêt !

Les aventures 2 à 9 se réfèrent toutes aux aventures qui les précèdent, il est donc recommandé de les suivre dans l'ordre. Si, bien entendu, tu préfères ne pas y consacrer l'attention nécessaire et les réaliser dans un autre ordre de ton choix, c'est tout à fait possible. Dans ce cas, tu pourras toujours chercher davantage d'aide sur un sujet particulier dans une aventure précédente s'il t'arrive de rencontrer quelque chose que tu ne connais pas.

En plus de t'aider à préparer ton ordinateur pour que tu puisses programmer ta carte Arduino, l'aventure 1 te guidera dans la réalisation de ton premier programme Arduino, qu'on appelle « sketch » ou « croquis » (traduction française du mot anglais *sketch*). À cette occasion, tu construiras ton premier circuit sur une plaque de montage pour contrôler une LED !

Dans l'aventure 2, tu apprendras à commander plus qu'une LED, à écrire un message depuis la carte Arduino vers ton ordinateur et à utiliser ton premier capteur, un

Copyright © 2015 Eyrolles.

potentiomètre. Tu pourras mettre en pratique les connaissances acquises en réalisant un panneau d'affichage où un message indique que tu ne souhaites pas être dérangé.

Dans l'aventure 3, tu prendras connaissance d'un deuxième capteur : un bouton-poussoir. Tu le combineras avec le potentiomètre pour contrôler un moteur. Tu apprendras également à utiliser les boucles `for` dans un programme d'ordinateur pour répéter plusieurs fois la même suite d'actions. En combinant tout cela, tu réaliseras un coffre-fort à combinaison qui ne s'ouvrira que lorsque le bon code sera entré. Le « coffre-fort » n'est qu'une boîte en carton, il ne conviendra donc pas pour les bijoux de ta mère, mais il sera assez utile pour y cacher tes bonbons favoris.

L'aventure 4 te montrera comment scinder ton code en petits sous-ensembles, en utilisant ce qu'on appelle les fonctions. Tu utiliseras ensuite ces fonctions pour contrôler plusieurs LED à l'aide d'un circuit intégré spécial appelé registre à décalage. Dans le projet final de cette aventure, tu fabriqueras des lettres dans lesquelles seront intégrées des LED, dans le style des banderoles des anciens carnavals.

L'aventure 5 ajoute le son à la liste grandissante d'actions que tu peux contrôler avec Arduino. Tu y découvriras comment créer des listes dans le code pour jouer des petits morceaux de musique à l'aide d'un nouveau composant : le transducteur piézoélectrique. Tu réaliseras alors un carillon à vent amplifié, qui ajoutera une déformation électronique sonore à un ancien instrument.

L'aventure 6 introduit des contrôles plus subtils qui te permettront de modifier la luminosité d'une LED au lieu de simplement l'allumer ou l'éteindre. Tu étendras également les possibilités de ton Arduino en installant de nouvelles bibliothèques de code qui ne sont pas fournies avec le logiciel Arduino. Tu utiliseras ensuite tes nouvelles connaissances avec une LED tricolore pour créer une boule de cristal qui change de couleur de façon magique.

Dans l'aventure 7, tu découvriras une nouvelle carte Arduino : la Leonardo. Tu apprendras à maîtriser une de ses possibilités passionnantes : se comporter comme un clavier d'ordinateur. Tu utiliseras un nouveau capteur qui détecte la lumière et réaliseras une manette de jeu avec laquelle tu pourras jouer en faisant des vagues avec ta main.

Dans l'aventure 8, tu utiliseras encore une autre carte Arduino, la LilyPad USB. Tu réaliseras un circuit en utilisant une aiguille et de la cordelette conductrice à la place du fil électrique et du fil à souder. Cette aventure te permettra de maîtriser les tableaux, dans ton code, et de créer un vêtement qui affiche un message secret.

Enfin, la plus grande des aventures sera l'aventure 9 ! Tu auras la chance d'y mettre à profit l'ensemble des compétences acquises durant les précédentes aventures pour créer un jeu de labyrinthe à bille qui calculera automatiquement ton score, décomptera le temps restant et produira des effets sonores. Tu emploieras un transducteur piézoélectrique pour détecter les vibrations et reproduire des effets sonores.

Copyright © 2015 Eyrolles.

Les annexes A et B te prépareront pour de nouvelles aventures au-delà de ce livre. L'annexe A t'orientera vers des livres et d'autres sources d'informations en ligne. L'annexe B t'indiquera où acheter les composants et les outils pour tes projets.

Présentation du texte

Tout au long de ce livre, tu rencontreras des encadrés prévus pour t'aider.

Ces encadrés expliquent les concepts ou les termes avec lesquels tu n'es pas familier.

Ces encadrés te donnent des astuces pour t'aider dans l'écriture du code et le montage des circuits.

Ces encadrés contiennent d'importantes mises en garde afin que ton ordinateur et toi-même soyez protégés au cours de l'exécution des étapes d'un projet.

Dans ces encadrés, un rapide questionnaire te permettra d'évaluer ta compréhension ou d'approfondir un sujet particulier.

Dans ces encadrés, je t'explique des choses ou te donne des informations supplémentaires qui te seront utiles.

Copyright © 2015 Eyrolles.

Ces encadrés contiennent des liens vers des vidéos (en anglais) qui te guideront dans les différentes étapes d'un projet.

Tu trouveras également deux autres types d'encadrés dans ce livre. L'encadré « Défi » te proposera de tirer profit de tes connaissances fraîchement acquises ou d'ajouter de nouvelles fonctions à tes projets. L'encadré « Exploration du code » te permettra de mieux comprendre les notions de programmation utilisées en codant Arduino.

En suivant les instructions du livre, tu devras entrer le code tel qu'il apparaît visuellement. Chaque « ; » est très important ! Par contre, les espaces entre les mots ne le sont pas : ils sont utilisés pour rendre le code plus lisible, mais ils ne sont pas importants pour Arduino. Par exemple, les deux lignes suivantes signifient la même chose :

```
if (i<4)
if ( i < 4 )
```

Ton sketch entier peut même être écrit sur une seule ligne de texte très longue, il fonctionnera quand même sur ton Arduino ! Ce sera cependant très difficile pour un autre programmeur de comprendre ce qui s'y passe. Il vaut mieux ajouter des espaces et des commentaires pour expliquer le contenu du programme : c'est la meilleure façon de coder.

Parfois, une ligne de code sera trop longue pour tenir sur une ligne du livre. Si tu vois le symbole ↩ à la fin d'une ligne de code, cela signifie que cette ligne et la suivante doivent être entrées sur une seule ligne de code dans le logiciel d'Arduino. Par exemple, les lignes suivantes doivent être entrées sur une seule ligne, pas sur deux :

```
Serial.println( "Hello, depuis la fonction setup() de ↩
ton Arduino Uno !" );
```

Pour que tu retiennes bien les notions de programmation expliquées dans ce livre, tu trouveras à la fin de chaque aventure un tableau qui liste les nouvelles commandes découvertes au cours du projet (fonctions, types de données, etc.).

Lorsque tu es arrivé à la fin d'une aventure, tu franchis une étape qui débloque un badge pour toi. Tu peux télécharger les badges de toutes les aventures en allant sur le site www.editions-eyrolles.com/go/arduino.

Le site web d'accompagnement

Tout au long du livre, tu trouveras des références au site web d'accompagnement du livre : www.editions-eyrolles.com/go/arduino

Copyright © 2015 Eyrolles.

Sur ce site, tu pourras télécharger le code (en français et en anglais) des programmes des différentes aventures. Ces programmes seront classés par aventures, et par projets s'il y en a plusieurs dans une aventure. Tu pourras aussi visionner des vidéos (en anglais) qui t'aideront dans la réalisation matérielle des projets et dans leur programmation.

Pour obtenir de l'aide

Dans l'annexe A de ce livre, tu trouveras de nombreux conseils pour te faire aider sur Arduino. Mais commence toujours par regarder le site d'Arduino (`www.arduino.cc`), car même s'il est en anglais, il contient de nombreuses informations dans la section d'apprentissage *Learning*. De plus, tu pourras toujours poser tes propres questions sur le forum, qui existe lui en français :
`https://forum.arduino.cc/index.php?board=33.0`

Un mot du traducteur

Les sites Internet et les ouvrages indiqués dans ce livre sont le plus souvent en langue anglaise. Toutefois, quand cela était possible, j'ai ajouté des sites ou des références en français qui traitent du même sujet. Tu pourras t'y rendre ou les consulter si tu ne peux pas lire l'anglais. Malheureusement, les sources d'informations les plus à jour et les plus complètes sont presque toujours dans la langue de Shakespeare... L'anglais technique étant généralement facile à lire, je t'invite à faire l'effort de parcourir ces documents. Je prends le pari, si tu as un niveau d'anglais scolaire du collège, que tu pourras rapidement comprendre les documents techniques dans cette langue. En plus, cela améliorera ton niveau en anglais.

Il est temps maintenant que tu partes à l'aventure !

Copyright © 2015 Eyrolles.

Aventure 1

Installe et teste ton Arduino

TU VAS BIENTÔT pouvoir créer des projets excitants qui feront le lien entre le monde physique et le monde digital ! Tu découvriras comment créer un programme qui déclenche un son, contrôle un moteur et fait clignoter la lumière. Arduino est l'outil parfait pour combiner des circuits et du code de programmation !

Tu suivras toujours les trois étapes suivantes, qui sont identiques pour chaque projet réalisé avec ton Arduino.

1. Écris sur ton ordinateur, à l'aide du logiciel Arduino, le code qui indique à ta carte Arduino Uno ce qu'elle devra faire.
2. Relie ta carte Arduino Uno à ton ordinateur pour y télécharger (téléverser) ton code.
3. Monte ton circuit puis relie-le à ta carte.

Commençons par le commencement. Avant de pouvoir faire quoi que ce soit, tu dois télécharger et installer le logiciel Arduino et préparer ton ordinateur pour programmer ta carte Arduino Uno. C'est ce que tu vas faire dans cette première aventure. Quand tu auras fait fonctionner le tout comme il se doit, tu pourras commencer ton premier projet Arduino : contrôler l'allumage et l'extinction d'une lampe.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

Pour commencer, tu as besoin des éléments suivants. La figure 1-1 montre les composants électroniques nécessaires :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une connexion Internet qui te permettra de télécharger le logiciel Arduino.

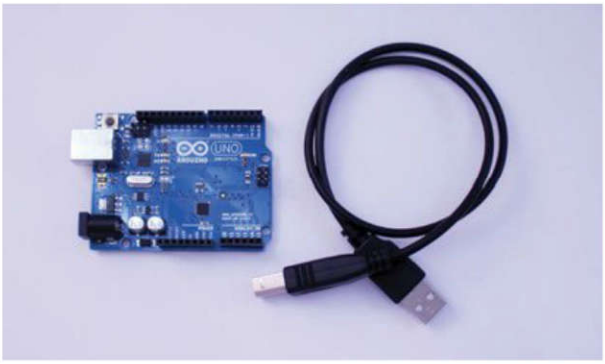


FIGURE 1-1 Le microcontrôleur Arduino Uno et son câble USB.

Téléchargement et installation du logiciel Arduino sur ton ordinateur

Pour pouvoir exécuter un programme Arduino, tu auras besoin, en plus de la carte Arduino Uno, d'un ordinateur et d'un logiciel spécial pour la faire fonctionner. Tu écriras d'abord sur un ordinateur le code qui va s'exécuter sur l'Arduino Uno, puis tu le téléchargeras dans ta carte. Cela te semble compliqué ? Ne t'inquiète pas, tu seras guidé pas à pas dans toutes les étapes. Et ce n'est pas aussi compliqué qu'il n'y paraît !

Tu vas utiliser un logiciel pour écrire le code de programmation, puis tu le téléchargeras. Ce logiciel s'appelle l'environnement intégré Arduino, ou ***Integrated Development Environment*** **(IDE)** en anglais. Tu le trouveras sur le site `http://arduino.cc/en/Main/Software` (voir la figure 1-2).

Copyright © 2015 Eyrolles.

FIGURE 1-2 Tu trouveras le logiciel IDE Arduino pour ton ordinateur sur le site web d'Arduino.

Un ***environnement de développement intégré*** **(IDE)** est un logiciel d'application utilisé pour écrire du code de programmation dans un langage particulier, on le nomme parfois « environnement de programmation ». Cette application permet de créer et modifier du code, mais aussi de le faire fonctionner (l'exécuter). De nombreux IDE offrent également des fonctions qui aident le programmeur à mettre au point (déboguer) leur programme. En d'autres termes, ils permettent de détecter les erreurs de programmation.

Tu vas maintenant télécharger et installer la dernière version du logiciel Arduino prévu pour le système d'exploitation de ton ordinateur en suivant les étapes ci-après. Quand ce livre a été écrit, la version courante de ce logiciel était la 1.6.5. Tu peux connaître la dernière version actuelle en te rendant sur le site `http://arduino.cc.en/Main/Software`. Tu y trouveras la dernière version en haut de la page. Après avoir installé le logiciel, tu pourras voir la version que tu utilises en lisant la barre du titre de la fenêtre de l'IDE Arduino.

Le logiciel peut fonctionner sur des ordinateurs Windows, Mac ou Linux. Toutefois, selon le système d'exploitation (*Operating Software* ou OS en anglais) utilisé, tu seras peut-être amené à installer un autre composant logiciel appelé **pilote**, en plus de l'IDE.

Copyright © 2015 Eyrolles.

Cette aventure décrit ce que tu dois faire pour installer le logiciel. Tu peux aussi te rendre sur le site `http://arduino.cc/en/Guide/HomePage`, qui, en anglais, donne de précieuses indications pour l'installer.

Garde ta carte Arduino et le câble USB à portée de main lors de l'installation du logiciel, car tu pourrais en avoir besoin à certaines étapes, cela dépendra du système d'exploitation de ton ordinateur.

Un **pilote** (*driver*, en anglais) est un petit logiciel qui permet à ton ordinateur de communiquer avec un dispositif externe comme une imprimante ou un clavier.

Installation du logiciel Arduino sur un Mac

Il est très facile d'installer le logiciel sur un Mac. Il ne sera pas nécessaire d'installer un pilote en plus de l'environnement Arduino. Voici les étapes à suivre.

1. Ouvre la page `http://arduino.cc/en/Main/Software` dans ton navigateur Internet.
2. Clique sur le fichier zip `Mac OS X`.
3. Cherche le fichier `arduino-1.6.5-r3-macosx.zip` (ou la version la plus récente) et décompresse-le en double-cliquant dessus. Déplace alors le fichier `Arduino.app` dans ton dossier d'applications.
4. Après avoir installé le logiciel, relie le câble USB à ta carte Arduino puis l'autre extrémité à ton ordinateur. Un message indiquant qu'un nouveau périphérique réseau est détecté pourra alors apparaître à l'écran. Si cela se produit, annule ou ferme la fenêtre du message.

Si tu as un problème quelconque, rends-toi sur le site `http://arduino.cc/en/Guide/MacOSX` pour trouver de l'aide supplémentaire.

Copyright © 2015 Eyrolles.

Pour voir une vidéo (en anglais) montrant l'installation de l'environnement Arduino sur un ordinateur fonctionnant sous Mac OS X, rends-toi sur le site d'accompagnement du livre www.editions-eyrolles.com/go/arduino.

FIGURE 1-3 Relie le câble USB à la carte Arduino, puis connecte l'autre extrémité à ton ordinateur.

Installation du logiciel Arduino sur un PC (Windows)

Tu dois suivre un certain nombre d'étapes pour installer le logiciel Arduino sous Windows. Tu devras en fait installer à la fois le logiciel et le pilote. Voici les étapes à suivre (il y aura peut-être quelques petites différences selon ta version de Windows, mais rien de bien méchant).

1. Dans ton navigateur Internet, rends-toi sur la page de téléchargement Arduino `http://arduino.cc/en/Main/Software`. Commence par télécharger le fichier nommé `arduino-1.6.5-r2-windows.zip`.
2. Cherche le fichier zip téléchargé et décompresse-le dans le dossier `Program Files`. Il devrait contenir plusieurs dossiers et fichiers, dont un dossier nommé `drivers` et un ficher nommé `arduino.exe`. Si tu veux, tu peux alors cliquer avec le bouton droit de ta souris sur le fichier `arduino.exe` et placer un raccourci vers ce fichier sur le Bureau de l'ordinateur.
3. Relie le câble USB à ta carte Arduino puis l'autre extrémité à ton ordinateur, comme le montre la figure 1-3. Tu devrais voir des voyants s'allumer sur ta carte Arduino Uno. Ceci indique simplement qu'elle est alimentée électriquement. Ton ordinateur va maintenant rechercher un pilote à utiliser avec la carte Arduino. Il se peut qu'il en trouve un de lui-même. Cependant, selon la version de Windows utilisée, tu peux avoir à suivre différentes étapes pour terminer l'installation du

Copyright © 2015 Eyrolles.

pilote. Tu auras peut-être besoin du mot de passe de l'administrateur de l'ordinateur et de l'aide de quelqu'un ayant une plus grande expérience des ordinateurs. Tu peux toujours te rendre sur le site `http://arduino.cc/en/Guide/Windows` pour trouver des instructions détaillées (en anglais). De nombreux sites français proposent également des tutoriels d'installation ou de démarrage tels que `http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.DebuterInstallationWindows`.

4. Clique sur le menu Démarrer et ouvre le Panneau de contrôle.
5. En étant dans le Panneau de contrôle, rends-toi dans Système et Sécurité. Clique sur Système et quand la fenêtre Système apparaît, ouvre le Gestionnaire de périphériques.
6. Dans le Gestionnaire de périphériques, regarde sous la rubrique Ports (COM & LPT) : tu devrais y voir un port nommé `Arduino Uno (COMxx)`. Si tu ne vois pas la section COM & LPT, regarde dans Autres périphériques et cherche `Périphérique inconnu`.
7. Clique avec le bouton droit sur le port `Arduino Uno (COMxx)` et choisis l'option Mise à jour du pilote.
8. Si tu utilises Windows XP ou si la fenêtre de recherche de pilotes apparaît, va sur le site `http://arduino.cc/en/Guide/UnoDriversWindowsXP` et suis les copies d'écran pour installer les pilotes situés dans le dossier que tu as téléchargé et décompressé.
9. Si la fenêtre de recherche de pilotes n'apparaît pas et si, à la place, une fenêtre avec l'option Rechercher automatiquement un pilote à jour et Rechercher un pilote sur mon ordinateur apparaît, clique sur Rechercher un pilote sur mon ordinateur.
10. Va dans le dossier que tu as téléchargé et décompressé, puis dans le dossier `drivers` qui s'y trouve, et choisis `arduino.inf`.

Wouah, c'était compliqué ! Heureusement, tu n'auras plus à le refaire : une seule fois suffit. Les étapes peuvent être modifiées lorsque de nouvelles versions de l'IDE Arduino apparaissent, ou s'il existe des mises à jour de ton système d'exploitation. Tu pourras toujours te rendre sur le forum français de ta carte Arduino en cas de problème d'installation, à l'adresse `https://forum.arduino.cc/index.php?board=33.0`. Tu pourras y lire les questions posées par d'autres utilisateurs et même poser tes propres questions, si tu ne trouves pas de sujets similaires aux problèmes que tu rencontres.

Copyright © 2015 Eyrolles.

Si tu rencontres des problèmes, visite d'abord le site http://arduino.cc/en/Guide/Windows pour obtenir plus d'aide. Le site http://mchobby.be/wiki/index.php?title=Guide_démarrage_Arduino en français pourra également t'apporter l'aide attendue.

Tu peux aussi visiter le site https://learn.adafruit.com/lesson-0-getting-started/installing-arduino-windows ou www.dummies.com/how-to/content/how-to-install-arduino-for-windows.html ou encore https://learn.sparkfun.com/tutorials/installing-arduino-ide/windows pour davantage d'astuces, dont un guide pour l'installation sous Windows 8.

Si ton ordinateur fonctionne sous Windows XP, tu peux installer les pilotes en suivant les instructions sur les copies d'écrans du site https://arduino.cc/en/Guide/UnoDriversWindowsXP.

Installation du logiciel Arduino sur un ordinateur Linux

Si ton ordinateur fonctionne sous Linux (par exemple si tu utilises un Raspberry Pi), tu devras d'abord consulter la documentation en ligne (en anglais) pour Linux et Arduino sur le site http://playground.arduino.cc/Learning/Linux. Nombreuses distributions de Linux existent, je ne les ai pas toutes citées ici. Si ton ordinateur fonctionne sous une de ces distributions, tu dois déjà savoir comment y installer un nouveau logiciel. Il te sera probablement facile d'installer l'environnement Arduino à partir d'un paquet (*package*, en anglais) en utilisant le gestionnaire de paquets. Voici comment réaliser l'installation sous une variante Debian de Linux, telle que Raspbian sur une carte Raspberry Pi. Sur la ligne de commande, entre la commande suivante :

```
sudo apt-get install arduino
```

Cette commande télécharge puis installe le logiciel. Tu peux aussi télécharger les versions 32 et 64 bits directement depuis la page de téléchargement http://arduino.cc/en/Main/Software. Tu n'auras pas à te soucier de la compilation du code source. Cependant, n'oublie pas de lire les instructions correspondant à la distribution que tu utilises à l'adresse http://playground.arduino.cc/Learning/Linux.

Si tu utilises un Raspberry Pi ou un autre système basé sur Linux, je suppose que tu sais comment utiliser la ligne de commandes pour installer des logiciels, quelle que soit la version de Linux que tu utilises. Si tu as besoin de plus d'informations ou de rafraîchir tes connaissances sur l'utilisation du Raspberry Pi, consulte l'ouvrage *À la découverte du Raspberry Pi* de Matt Richardson et Shawn Wallace (éditions Eyrolles, 2014).

Copyright © 2015 Eyrolles.

À la découverte de l'IDE Arduino

Bien joué ! Maintenant que tu as installé le logiciel d'environnement Arduino, tu es prêt à utiliser l'IDE Arduino ! Il est temps de lancer le logiciel. Si ton ordinateur est un Mac, va dans le dossier Applications et double-clique sur Arduino. Si ton ordinateur fonctionne sous Windows, double-clique sur le raccourci que tu as créé sur le Bureau (si tu en as créé un comme indiqué précédemment dans cette aventure), ou va dans le dossier que tu as téléchargé précédemment et double-clique sur l'application Arduino. Si ton ordinateur fonctionne sous Linux, lance l'IDE Arduino depuis la ligne de commandes ou en double-cliquant sur l'icône du programme. L'icône du programme Arduino (voir la figure 1-4) est le même pour Mac, Windows ou Linux.

Lorsque l'IDE Arduino a terminé sa phase de lancement, une fenêtre similaire à celle de la figure 1-5 apparaît.

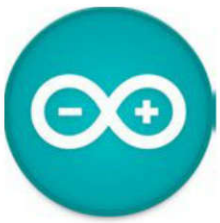


FIGURE 1-4 L'icône du logiciel Arduino.

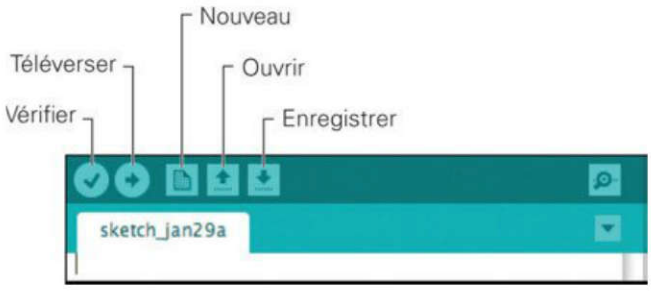


FIGURE 1-5 La partie importante de l'IDE Arduino.

Par défaut, le logiciel Arduino possède des menus en anglais. Pour le faire fonctionner en français, effectue les opérations suivantes :

- après avoir lancé l'IDE Arduino, choisis le menu File puis Preferences avant de sélectionner Français (French) dans la case Editor language ;
- clique ensuite sur OK ;
- redémarre ensuite le logiciel Arduino qui devrait maintenant fonctionner en français.

Pour commencer, tu vas découvrir quelques-unes des fonctions de base de l'IDE Arduino. Il y a six boutons en haut de la fenêtre (voir la figure 1-5), le rôle de la plupart d'entre eux est facile à deviner. Les trois boutons du centre sont Nouveau, Ouvrir et Enregistrer. En cliquant sur Nouveau (surprise !), tu crées un nouveau fichier Arduino. Le bouton Ouvrir permet d'ouvrir un fichier Arduino existant. Enfin, Enregistrer permet de sauvegarder le fichier en cours. Tu remarqueras qu'il y a aussi un bouton isolé à l'extrémité droite de la fenêtre : tu t'en serviras au cours de l'aventure 2. Il ne te reste plus que les boutons de gauche à découvrir.

Copyright © 2015 Eyrolles.

Le bouton qui ressemble à une case cochée est le bouton de vérification. Quand tu cliques dessus, le logiciel Arduino **compile** le code de ton programme : autrement dit, il prend le code que tu as écrit et le traduit en quelque chose de compréhensible par la carte Arduino. Une seconde action peut être déclenchée à l'aide de ce bouton. S'il y a des parties de code que le logiciel Arduino ne comprend pas, par exemple un point virgule manquant ou une erreur typographique durant la saisie des instructions, un message d'erreur est affiché en bas de la fenêtre de l'IDE Arduino. Celui-ci tente d'être explicite dans son message, mais ce dernier n'a parfois pas de sens ! Plus loin, dans cette aventure, tu verras les messages d'erreurs les plus fréquents et ce qu'ils signifient.

Compiler le code est un processus qui prend le code écrit par l'être humain et le transforme en instructions qui peuvent être comprises par une machine.

Ne te soucie pas encore trop de la façon dont cela fonctionne. Tout ce que tu dois retenir pour l'instant, c'est que le bouton Vérifier est utilisé pour traduire ton code en quelque chose que Arduino peut comprendre, et aussi pour savoir si le code contient des erreurs simples. Après avoir vu cela, clique sur le bouton Téléverser. Ce bouton compile et télécharge (téléverse) le code dans l'Arduino Uno. Le bouton Vérifier peut être utile quand tu écris un programme et que tu ne disposes pas de la carte Arduino Uno près de toi, car cela permet au moins de vérifier que ton code ne contient pas d'erreurs simples. En revanche, tu ne peux pas savoir si ton code fonctionne correctement avant de l'avoir téléversé dans ta carte Arduino.

Téléverser le code est un processus qui prend le code traduit sous une forme comprise par Arduino et le transmet à la carte Arduino qui va pouvoir le lancer.

Vérification de l'installation avec le test du clignotement

D'habitude, tu peux écrire un programme pour un ordinateur comme un portable ou un Raspberry Pi, puis le lancer sur le même appareil que celui sur lequel tu l'as écrit. Avec Arduino, c'est légèrement différent, car une carte Arduino ne peut pas se programmer elle-même. En effet, tu dois d'abord écrire le code Arduino sur un ordinateur qui peut exécuter l'IDE Arduino. Le code qu'un être humain peut écrire est lu par l'IDE

Copyright © 2015 Eyrolles.

puis traduit (ou compilé) en un code que la carte Arduino comprend. L'IDE télécharge ensuite le code compilé sur la carte Arduino.

Les programmeurs utilisent des petits programmes simples pour vérifier que leur ordinateur fonctionne tel qu'ils le souhaitent. Ces petits programmes sont appelés les programmes *Hello World* par les Anglo-Saxons, ou *Bonjour tout le monde* en français.

Un programme *Hello World* est un programme très simple. Si tu apprends un nouveau langage de programmation, tu écriras probablement un programme qui écrira simplement la phrase « Hello World » (ou « Bonjour tout le monde ») sur l'écran. Mais attention, Arduino n'a pas d'écran ! Après avoir téléchargé le code de ton ordinateur vers la carte Arduino, il n'y a plus de communication avec l'ordinateur ! Le câble USB ne sert plus qu'à alimenter le microcontrôleur (tu verras cependant, dans la prochaine aventure, comment envoyer des messages à l'ordinateur à l'aide du câble USB).

La carte Arduino peut faire clignoter une lampe. Elle en possède même une qui est prévue pour cela. Une petite **LED** (diode électroluminescente ou *light emitting diode* en anglais) orange ou rouge se trouve près du nombre 13 inscrit sur la carte (voir la figure 1-6). Cette LED est utile, car tu peux la programmer pour l'allumer puis l'éteindre afin de vérifier rapidement que ta carte Arduino fonctionne. Tu n'as besoin de rien en dehors de ta carte Arduino pour vérifier que tout fonctionne comme prévu.

L'IDE Arduino comprend des exemples de code pour t'aider à débuter. L'un de ces exemples est le programme (également appelé **sketch**) de clignotement qui, pour Arduino, est utilisé à la place d'un programme *Bonjour tout le monde*. Tu vas maintenant l'utiliser pour vérifier que tout fonctionne comme tu veux.

Une **LED**, ou **diode électroluminescente**, est un composant électronique qui s'illumine quand un courant électrique la traverse. Une diode ne laisse passer le courant électrique que dans un sens. Une LED laisse passer le courant seulement quand son fil le plus long est relié au pôle positif d'une source d'alimentation électrique et son fil le plus court au pôle négatif. Si les fils sont inversés, elle reste éteinte.

Les programmes Arduino sont souvent appelés **sketches**, ou encore **croquis**, comme les petits dessins que font les artistes.

Copyright © 2015 Eyrolles.

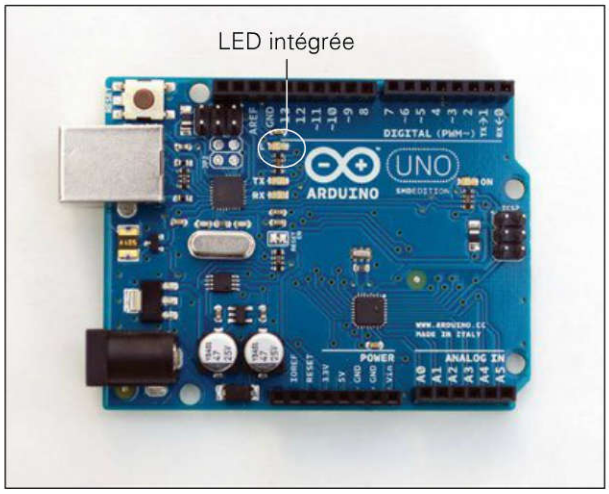


FIGURE 1-6 La diode LED intégrée sur la carte Arduino près du nombre 13.

Téléchargement du code de clignotement

Pour vérifier que ton Arduino Uno peut recevoir des messages et un nouveau programme de ton ordinateur, tu dois compiler et télécharger le sketch de clignotement (*Blink sketch*). Va dans le menu Fichier > Exemples > Basics > Blink (voir la figure 1-7) afin d'ouvrir une nouvelle fenêtre contenant le sketch `Blink`.

Avant de télécharger ton code, tu dois vérifier deux réglages : le type de carte et le port de communication. Tu dois t'assurer que ces paramètres sont corrects chaque fois que tu démarres l'IDE Arduino. Après les avoir vérifiés, tu ne devras plus les changer jusqu'à ce que tu quittes et redémarres l'IDE à nouveau. Pour trouver le type de carte, choisis Outils > Type de carte (voir la figure 1-8). Tu devrais y trouver une liste de toutes les cartes Arduino. Assure-toi de choisir celle que tu utilises. (Ce sera normalement une Arduino Uno, mais si tu n'es pas sûr, le modèle est écrit sur la carte).

Tu trouveras le port dans le menu Outils > Port (voir la figure 1-9). Choisis le port que tu utilises de la même façon que tu as choisi le type de carte. Assure-toi que ta carte Arduino est reliée à ton ordinateur avec le câble USB, sinon le port n'apparaîtra pas dans la liste.

Copyright © 2015 Eyrolles.

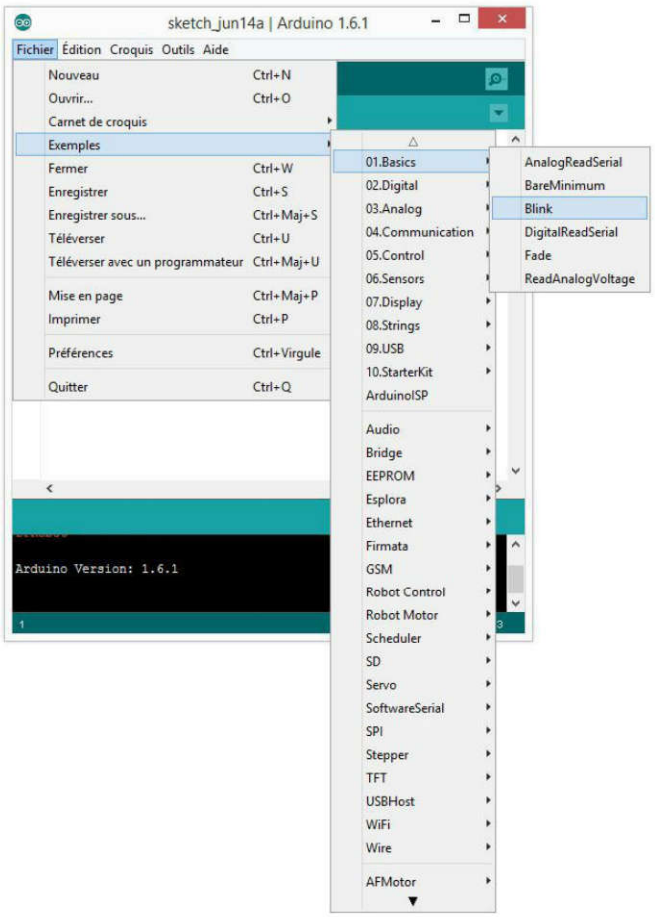


FIGURE 1-7 Ouverture du sketch Blink qui se trouve dans les exemples inclus dans l'IDE Arduino.

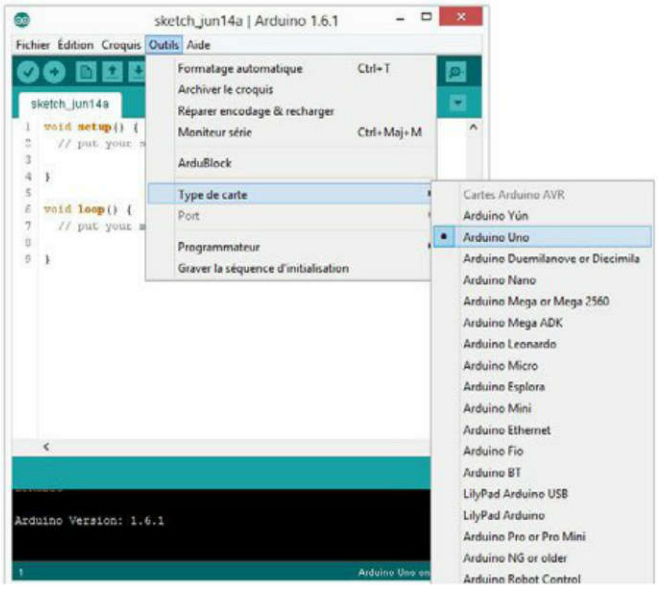


FIGURE 1-8 Sélection du type de carte utilisée.

Copyright © 2015 Eyrolles.

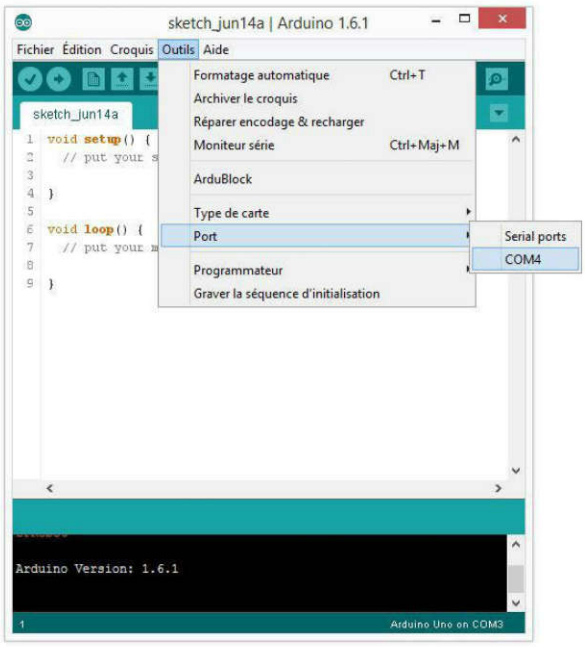


FIGURE 1-9 Sélection du port auquel est reliée ta carte Arduino.

Il peut être difficile de déterminer quel port tu dois choisir quand plusieurs ports sont indiqués dans la liste du menu. Tu peux consulter cette liste avant et après que la carte ne soit reliée. Si un port apparaît après que la carte est connectée, alors c'est celui-ci que tu devras choisir.

Si tu utilises un Mac, cherche le port qui commence par `/dev/tty.usbmodem`. Sur un ordinateur Windows, le port est simplement indiqué par `COM`. Sous Linux, le port commence par `/dev/ttyACM`. Il n'y a aucun risque si tu ne choisis pas le bon port. Si tu n'es pas sûr, choisis un port puis clique sur le bouton Téléverser. Si tu obtiens un message d'erreur, choisis le port suivant dans la liste et continue ainsi jusqu'à ce que tu trouves celui auquel ta carte Arduino est reliée.

Rappelle-toi que tu dois relier à l'ordinateur le câble USB qui est connecté à ta carte Arduino, sinon le port ne sera pas dans la liste des ports disponibles.

Copyright © 2015 Eyrolles.

Après avoir choisi la carte et le port, tu es prêt à téléverser ton code : le croquis Blink que tu as ouvert. Pour cela, clique simplement sur le bouton Téléverser. Si tu as oublié de choisir le port, un message apparaîtra te demandant d'en choisir un.

Si tu as tout correctement préparé, tu verras un message dans la fenêtre du bas de l'IDE Arduino t'informant de la compilation du croquis puis de son téléchargement. Si le sketch est téléchargé sans problème, tu verras apparaître un message de fin (en caractères blancs).

FIGURE 1-10 Le message de l'IDE Arduino après le téléchargement correct d'un sketch.

Regarde la LED à côté du nombre 13 sur ta carte. S'allume-t-elle pendant une seconde puis s'éteint-elle pendant une seconde ? Si oui, alors bravo, car tu as réussi ! Bien joué ! Si non, ou si un message en caractères orange est apparu dans la fenêtre du bas de l'IDE, la section suivante va t'aider à déterminer ce qui ne s'est pas bien passé.

Recherche des problèmes courants

Lorsque quelque chose ne se passe pas correctement quand tu télécharges le code sur la carte Arduino, un message provenant d'un module appelé `avrdude` est affiché dans la fenêtre du bas de l'IDE (comme celui de la figure 1-11).

La carte Arduino est réalisée autour d'un circuit intégré microcontrôleur (calculateur) fabriqué par la société Atmel. Ce circuit intégré est un microcontrôleur de type AVR et le logiciel que l'IDE Arduino utilise pour communiquer avec la carte Arduino est appelé avrdude. Ainsi, quand tu obtiens des messages provenant d'`avrdude`, cela signifie que quelque chose s'est mal passé lors de la communication entre l'ordinateur et la carte. Généralement, c'est parce que l'ordinateur essaie d'envoyer un nouveau sketch à la carte

Copyright © 2015 Eyrolles.

Arduino Uno par l'intermédiaire d'avrdude, mais l'ordinateur ne la trouve pas. Ces problèmes peuvent être dus au choix d'un mauvais port. Cependant, si tu as bien essayé tous les ports, il peut s'agir d'un autre problème.

FIGURE 1-11 Une erreur courante quand l'ordinateur ne parvient pas à communiquer avec Arduino.

Le plus simple à faire, quand tu obtiens une erreur d'avrdude, est de déconnecter ta carte Arduino de son câble USB (cela supprime son alimentation électrique et la met à l'arrêt) puis de la reconnecter. Si tu as toujours des problèmes, essaie de déconnecter la carte Arduino, puis arrête l'IDE Arduino ainsi que tout autre programme. Redémarre ensuite l'IDE puis reconnecte l'Arduino Uno pour voir si tu réussis à télécharger un nouveau sketch.

Si tu as fait tout cela et que tu ne parviens toujours pas à télécharger des sketches sur ta carte Arduino, suis à nouveau toutes les étapes de l'installation de l'IDE et des pilotes.

Le site web d'Arduino (http://arduino.cc) est une aide précieuse renfermant de nombreux tutoriels. Il contient également un forum en français où tu pourras poser des questions. Tu y trouveras probablement des questions déjà posées par d'autres personnes ayant eu le même problème que toi.

EXPLORATION DU CODE

Par chance, tu as maintenant téléversé et fait fonctionner ton code sur ta carte Arduino Uno.

Mais que fait réellement le code ? Tu sais que l'Arduino Uno allume puis éteint la LED située près du nombre 13, mais comment sait-elle qu'elle doit le faire ?

Copyright © 2015 Eyrolles.

Une manière efficace de commencer l'étude d'un programme consiste à le lire avant de l'écrire. Après tout, à l'école, tu n'as pas appris à écrire avant de savoir lire ! Tu peux utiliser le sketch Blink comme introduction au code. Ne te soucie pas d'en connaître tous les détails dès maintenant, car il y a beaucoup de choses à apprendre. Tu découvriras tout un tas de nouveaux termes, mais tu n'auras pas besoin de retenir ce qu'ils signifient pour le moment. Tu consacreras plus de temps à les comprendre dans les aventures suivantes.

Si tu regardes à nouveau le sketch Blink, tu verras que la première partie est en gris. L'IDE Arduino aide à comprendre ce qui se passe dans un programme en changeant les couleurs du code selon ce qu'il fait. Le texte qui est en gris s'appelle un **commentaire**. Les commentaires sont des notes à l'attention des programmeurs pour les aider à comprendre ce qui se passe dans le code. Le long commentaire au début du sketch explique ce qu'il fait.

Les **commentaires** sont des notes à l'intérieur d'un programme qui expliquent ce qu'une ligne ou une partie de code fait. Chaque ligne de commentaire commence par `//`. Si tu veux écrire un commentaire de plusieurs lignes, il commencera par `/*` et se terminera par `*/`. Ces caractères spéciaux indiquent à l'ordinateur exécutant le programme d'ignorer la ou les ligne(s) de commentaires.

La version originale en anglais du sketch Blink :

```
/*
  Blink
  Turns a LED on for one second, then off for
  one second, repeatedly.

  Most Arduinos have an on-board LED you can
  control. On the Uno and the Leonardo, it is
  attached to digital pin 13.
  If you're unsure what pin the on-board LED is
  connected to on your Arduino model, check the
  documentation at http://arduino.cc.

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald
*/
```

Copyright © 2015 Eyrolles.

La version francisée :

```
/*
  Blink
  Allume une LED pendant une seconde, puis l'éteint
  pendant une seconde, de façon répétitive.

  La plupart des cartes Arduino possèdent une LED intégrée
  que tu peux contrôler. Sur la Uno et la Leonardo, elle
  est reliée à la broche numérique 13.
  Si tu n'es pas sûr de la broche à laquelle la LED est
  reliée, vérifie la documentation de ta carte sur le site
  http://arduino.cc.

  Cet exemple de programme est dans le domaine public.

  Modifié le 8 mai 2014
  par Scott Fitzgerald
*/
```

Les lignes qui ne commencent pas par `//` sont des lignes de code que l'ordinateur doit exécuter. Celles qui commencent par `//` sont ignorées par l'ordinateur, ce sont des notes expliquant ce que fait le code.

Chaque sketch Arduino doit posséder deux **fonctions** : `setup()` et `loop()`. Une fonction est un ensemble de lignes de code qui possède un nom. La portion suivante du code est la fonction `setup()`. Cette fonction ne s'exécute qu'une seule fois, elle est prévue pour les tâches qui ne doivent être réalisées qu'une fois au démarrage d'Arduino. Chaque fois que la carte démarre, elle cherche la partie du sketch contenant la fonction `setup()` et commence par exécuter le code qu'elle contient.

Une **fonction** est un ensemble de lignes de code qui possède un nom. Une fonction peut être réutilisée de façon répétitive. Elle peut utiliser des informations qui lui sont fournies en entrée et délivrer des informations en sortie lorsqu'elle a terminé, mais toutes les fonctions n'ont pas à le faire.

Copyright © 2015 Eyrolles.

La version originale en anglais :

```
/* The setup function runs once when you press
  reset or power the board. */
  void setup() {
// initialize digital pin 13 as an output
  pinMode(13, OUTPUT);
}
```

La version francisée :

```
/* La fonction setup est exécutée quand tu
  appuies sur le bouton reset ou alimente la
  carte. */
  void setup() {
// initialise la broche 13 en mode sortie
  pinMode(13, OUTPUT);
}
```

Comme tu peux le voir, il y a une seule instruction à l'intérieur de la fonction `setup()` : c'est un appel à la fonction `pinMode()`. Chaque broche d'Arduino peut lire des informations ou en délivrer en sortie, mais elle ne peut pas le faire de façon simultanée. La fonction `pinMode()` indique si la broche est utilisée en entrée ou en sortie grâce à deux **arguments**. Le premier est le numéro de la broche que tu utilises. Le second est un mot-clé spécial (`OUTPUT`), qui indique à Arduino que tu souhaites utiliser la broche 13 en sortie et non pas en entrée.

Un **argument** est une information transmise à une fonction, que celle-ci utilise pour exécuter sa tâche. L'argument se place entre les parenthèses qui suivent le nom de la fonction. Dans le fragment de code suivant, par exemple, la fonction `delay(1000)` a comme argument 1000 : c'est le nombre de millisecondes pendant lesquelles on veut qu'Arduino attende avant d'exécuter la ligne de code suivante.

Le reste du code du sketch Blink est la fonction `loop()`. Après que l'Arduino Uno exécute le code contenu dans `setup()`, elle cherche la fonction appelée `loop()` et exécute alors l'ensemble du code qu'elle contient. Quand elle a terminé, elle exécute à nouveau le code de cette même fonction. Puis encore

Copyright © 2015 Eyrolles.

à nouveau ! Et ainsi de suite ! La fonction `loop()` se répète indéfiniment (ou au moins jusqu'à ce que Arduino ne soit plus alimentée électriquement).

La version originale en anglais :

```
/* The loop routine runs over and over again
  forever */
void loop() {
  digitalWrite(13, HIGH); /* turn the LED on
  (HIGH is the voltage level) */
  delay(1000); // wait for a second
  digitalWrite(13, LOW); /* turn the LED off by
  making the voltage LOW */
  delay(1000); // wait for a second
}
```

La version francisée :

```
/* La fonction loop s'exécute de façon
  répétitive */
void loop() {
  digitalWrite(13, HIGH); /* allume la LED
   (HIGH est le niveau de tension HAUT) */
  delay(1000); // pause d'une seconde
  digitalWrite(13, LOW); /* éteint la LED en
   mettant le niveau de sortie BAS (LOW)*/
  delay(1000); // pause d'une seconde
}
```

La première ligne de code dans la fonction `loop()` allume la LED en utilisant la fonction `digitalWrite()`. Cette fonction reçoit deux arguments : le numéro de broche, et si oui ou non tu délivres un signal électrique sur la broche. L'argument du numéro de broche est identique à celui de la fonction `pinmode()`. Le second argument est un mot-clé, `HIGH` ou `LOW`. `HIGH` délivre une tension électrique en sortie alors que `LOW` n'en délivre pas.

La dernière partie du code, que tu rencontres pour la première fois, est la fonction `delay()`. Arduino exécute son code très rapidement, des millions de fois par seconde. C'est si rapide que tu ne pourrais pas voir la diode LED s'allumer et s'éteindre. Tu dois donc demander à ta carte de faire des pauses pour que tu puisses voir la LED clignoter. La fonction `delay()` lui commande d'attendre le nombre de millisecondes qui lui est fourni comme argument. Dans cet exemple, le délai est de 1 000 millisecondes, ce qui est égal à une seconde. Voici un résumé de ce que fait l'ensemble du croquis :

- la fonction `setup()` utilise la fonction `pinmode()` pour indiquer que la broche à laquelle est reliée la LED est utilisée en sortie ;

Copyright © 2015 Eyrolles.

- dans la fonction `loop()`, la sortie de la broche 13 délivre une tension électrique à la LED ;

- Arduino fait ensuite une pause de 1 000 millisecondes ;

- puis la sortie de la broche de la LED ne délivre plus de tension ;

- et Arduino fait à nouveau une pause de 1 000 millisecondes ;

- ensuite, la fonction `loop()` s'exécute à nouveau.

Un dernier détail que tu as peut-être remarqué : c'est l'étrange ponctuation utilisée dans le code. La plupart des lignes se terminent par un point-virgule (`;`), comme un point à la fin d'une phrase. Un point indique qu'une phrase est terminée, ce qui permet une écriture nette et propre. Un point-virgule signifie la même chose pour un ordinateur : il permet à l'ordinateur de séparer les différentes lignes de code.

Tu as peut-être noté également que le code contient des espaces et des indentations (des retraits de textes). De même que les commentaires, ils permettent au code d'être plus facile à comprendre par le programmeur. L'ordinateur les ignore. L'indentation permet de voir facilement les lignes de code appartenant à une même fonction. Tu as sans doute remarqué que les lignes de code appartenant à la fonction `loop()` sont toutes situées entre les crochets { et } et sont indentées (en retrait). Les crochets { } indiquent à l'ordinateur que ces lignes appartiennent à la fonction `loop()`, alors que l'indentation rend cela visuellement plus clair.

Les espaces entre les parties du code sont également prévus pour le programmeur, l'ordinateur les ignore aussi. Par exemple, `delay(1000)` et `delay( 1000 )` sont identiques pour l'ordinateur, mais les espaces permettent une meilleure lisibilité du code pour certaines personnes. Tu pourras décider comment tu préfères écrire ton code dans les aventures suivantes !

DÉFI

Essaie de modifier la durée pendant laquelle la LED s'allume puis s'éteint en changeant les arguments des fonctions `delay()`. Rappelle-toi que tu dois téléverser ton code à chaque fois que tu le modifies afin que le sketch avec les nouveaux changements soit bien dans Arduino.

Copyright © 2015 Eyrolles.

Réalisation d'un circuit avec une LED

Maintenant que tu as une LED clignotante sur ta carte Arduino Uno, tu es prêt à aller plus loin en construisant ton premier circuit ! Tu vas d'abord te familiariser avec des outils dont tu auras besoin pour réaliser et tester de nouveaux circuits : les schémas et les plaques de montage. Ce sont les éléments essentiels pour réaliser tes propres projets que tu pourras ensuite partager, et aussi pour être capable de construire les projets créés par d'autres. Dans l'aventure suivante, tu franchiras une étape supplémentaire en construisant un boîtier pour ton circuit. Mais d'abord, familiarise-toi avec la façon dont le code et les circuits se combinent avec l'Arduino Uno.

Ce dont tu as besoin

Tu dois disposer des éléments suivants pour construire ton circuit à LED. La figure 1-12 montre les composants électroniques nécessaires :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- une résistance de 220 Ω ;
- une LED ;
- 2 fils de pontage.

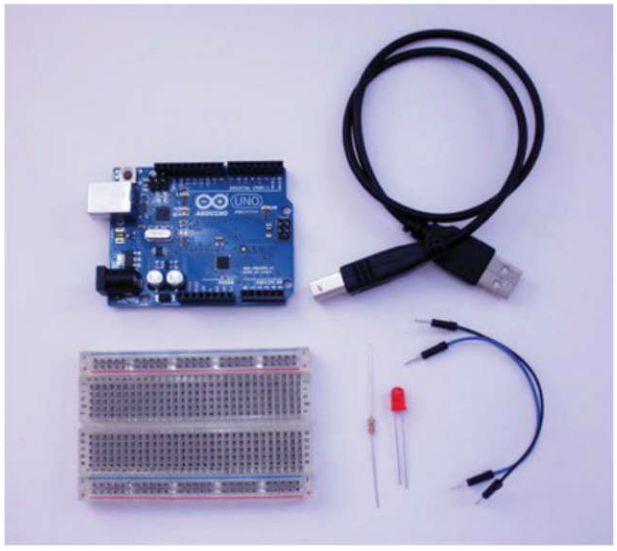


FIGURE 1-12 Les composants électroniques dont tu as besoin pour réaliser ton circuit.

Copyright © 2015 Eyrolles.

Compréhension des schémas des circuits

L'électricité est un flux de charges électriques. Tu l'as vu dans la nature avec les orages et avec l'électricité statique qui survient quand tu marches sur un sol en moquette puis que tu touches une poignée de porte. Tu utilises aussi des circuits tous les jours pour contrôler la circulation de l'électricité. Par exemple, tu allumes et éteins l'éclairage d'une pièce à l'aide d'un interrupteur. Tu peux aussi allumer la télévision et changer de chaîne. Tout cela est réalisé à l'aide de circuits qui contrôlent l'électricité. Tu n'es pas encore prêt à réaliser des circuits comme ceux qui sont à l'intérieur d'un téléviseur, mais tu peux en construire un qui allume et éteint une lampe !

Même très simple, un circuit peut être réalisé de différentes manières. Par exemple, les LED existent en plusieurs tailles et couleurs. Tu peux alimenter une LED depuis l'Arduino Uno ou en utilisant une pile. Le schéma d'un circuit est un diagramme simplifié qui montre les informations importantes du montage en utilisant des symboles. La figure 1-13 montre le schéma du circuit que tu vas réaliser. Ce schéma contient trois symboles qui représentent l'Arduino Uno, la résistance (on verra cela plus tard) et la LED (le symbole triangulaire). Tu pourrais réaliser ce circuit avec une LED rouge de grande taille ou avec une petite diode de couleur verte, c'est ton choix. L'information importante est que tu dois relier cette LED à l'Arduino Uno et à une résistance.

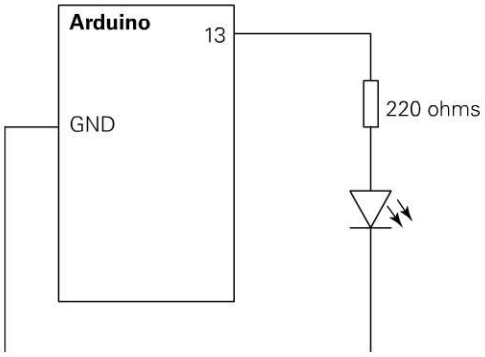


FIGURE 1-13 Le schéma du circuit à LED.

L'électricité peut être définie et mesurée de plusieurs façons. Parce qu'elle est invisible, il est difficile d'imaginer comment l'électricité agit, c'est pour cela que l'on utilise souvent une analogie avec l'eau. Le flux du courant électrique dans un fil est comparable à l'eau circulant dans un tuyau. L'eau s'écoulant au travers du tuyau est similaire au courant électrique (qui est mesuré en ampères et que l'on symbolise avec la lettre A) et la pression de l'eau est comme la tension électrique (qui est mesurée en volts et symbolisée par la lettre V). Le diamètre du tuyau combiné avec la quantité d'eau qui doit le traverser modifie la pression de l'eau. Un tuyau plus petit crée une pression d'eau supérieure à un gros tuyau pour la même quantité d'eau à faire passer. Le diamètre du tuyau représente la troisième propriété utilisée pour décrire l'électricité : la résistance électrique (mesurée en ohms et représentée par le symbole Ω).

Copyright © 2015 Eyrolles.

Cela peut paraître compliqué et difficile à comprendre, mais n'essaie pas d'en saisir tous les détails dès maintenant. En travaillant avec l'électricité et en réalisant des circuits, tu verras comment tension, courant et résistance sont en relation. Revenons donc au schéma du circuit !

L'étude de l'électricité et des circuits est appelée la théorie des circuits. Si tu es intéressé par un apprentissage plus approfondi de cette théorie, il existe de nombreux tutoriels en ligne qui te permettront d'en savoir plus. J'aime beaucoup ceux de SparkFun (en anglais). Ils contiennent des animations qui aident à illustrer les concepts ! Commence par `https://learn.sparkfun.com/tutorials/what-is-electricity` et `https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law`.

Dans le schéma de la figure 1-13, le circuit est relié à la broche 13 et à la broche GND (masse) d'Arduino. La broche 13 est celle depuis laquelle l'électricité qui va allumer la LED circule. GND signifie la Terre, ou la Masse, ou le potentiel 0 V. Si tu utilises une pile pour alimenter un circuit, celle-ci possède deux connexions, l'une positive et l'autre négative. L'électricité circule du pôle positif vers le pôle négatif. Les broches 13 et GND de la Uno ont exactement le même rôle que le positif et le négatif d'une pile. L'électricité circule de la broche 13 vers la masse (GND) quand elles sont reliées à un circuit.

Le circuit contient deux composants en plus de la carte Arduino Uno : une résistance et une LED. Toutes les broches de la carte fournissent une tension de 5 V ce qui est trop pour une LED. Souviens-toi de l'analogie avec l'eau et imagine que la LED est une paille pour boire. Elle n'est pas aussi résistante qu'un tuyau d'eau, et si l'on force le passage de trop d'eau au travers de la paille, cela pourrait la faire éclater. Une **résistance** aide à contrôler le courant qui circule au travers de la LED. Dans ce type de circuit, la résistance limite la quantité de courant, c'est pourquoi on l'appelle « résistance de limitation de courant ».

Une **résistance** est un composant électrique qui résiste au passage du courant dans un circuit. Par exemple, une LED pourrait être endommagée par un courant trop fort, mais si tu ajoutes une résistance de valeur correcte au circuit pour limiter le courant qui le traverse, la LED est alors protégée. La résistance se mesure en ohms et est représentée par le symbole Ω. Tu dois choisir une résistance de valeur correcte pour limiter le courant au travers d'un circuit. La valeur de la résistance est indiquée par des anneaux de couleur qui se lisent de gauche à droite.

Copyright © 2015 Eyrolles.

Utilisation d'une plaque de montage

Maintenant que tu connais le circuit que tu vas réaliser, tu as besoin d'un élément appelé **plaque de montage** pour t'aider à le construire. Une plaque de montage est une plaque qui possède des rangées de trous (voir la figure 1-14). Si tu pouvais voir l'intérieur d'une telle plaque, tu verrais que les trous de chaque rangée font face à la même partie métallique. Relier deux fils dans la même rangée signifie qu'ils sont reliés à la même partie métallique et qu'un courant électrique peut donc les traverser.

Une **plaque de montage** est un composant réutilisable qui permet de créer des circuits sans avoir besoin de souder les composants. Une plaque de montage possède une grande quantité de trous dans lesquels on peut insérer des fils et des composants formant un circuit.

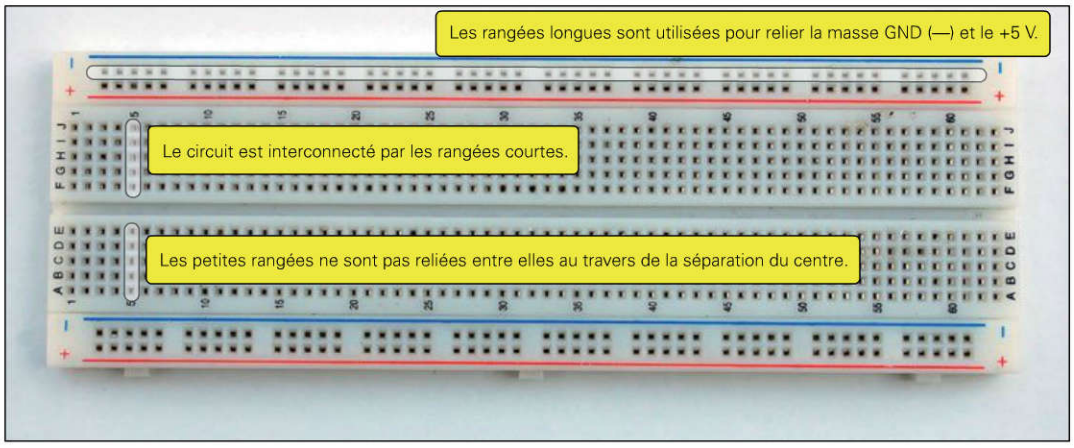


FIGURE 1-14 Une plaque de montage possède une série de trous qui sont reliés par rangées, dont deux paires de rangées longues près des bords et des rangées perpendiculaires, plus courtes, situées au centre de la plaque.

Les rangées les plus longues, situées sur le bord externe de la plaque, sont celles où tu dois connecter le +5 V et la masse (ou 0 V, GND sur la carte Arduino). Certaines plaques peuvent comporter des inscriptions comme + ou –, ou des couleurs comme du rouge avec du noir ou du bleu. Le rouge représente la tension positive, qui est de 5 V dans le cas de la Uno ; le noir (ou le bleu) représente sa masse GND.

Tu peux considérer les deux rangées longues pour le +5 V et GND comme une extension des broches +5 V et GND de la Uno. Il n'y a qu'une broche +5 V sur l'Arduino Uno, alors que se passera-t-il, si tu as plus d'un composant qui doit y être relié ? En utilisant un fil de pontage, tu pourras relier la broche +5 V à une rangée de la plaque de montage. Tu disposeras ainsi de beaucoup de trous où tu pourras connecter des composants à

Copyright © 2015 Eyrolles.

cette broche. Il en est de même pour la masse GND. Il y a davantage de broches GND sur l'Arduino Uno (il y en a trois et elles sont identiques, tu peux utiliser n'importe laquelle pour relier les composants à la masse), mais tu peux aussi utiliser un fil de pontage pour augmenter le nombre de trous de connexion à la masse en le connectant à la plaque de montage. Cette façon de faire est représentée sur la figure 1-15.

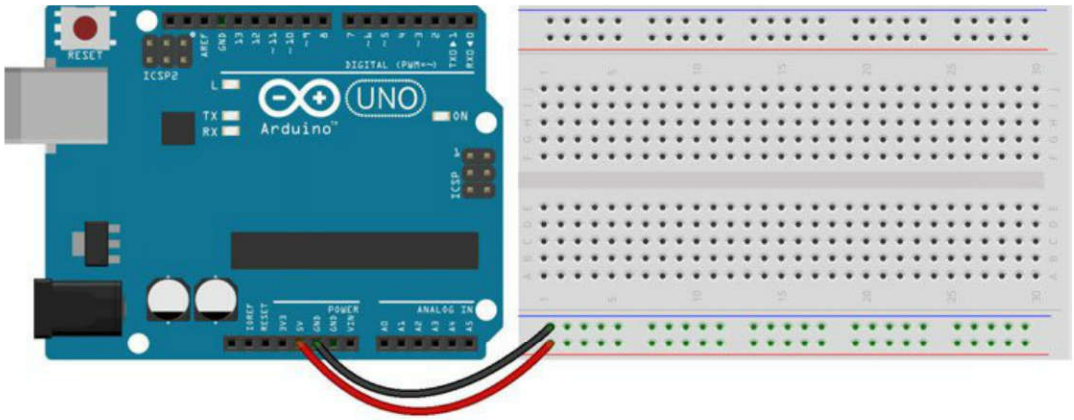


FIGURE 1-15 Connexion des lignes d'alimentation d'une plaque de montage.

Les rangées situées au milieu de la plaque de montage, entre les grandes rangées des bords, sont celles que tu dois utiliser pour relier tes composants entre eux. Si tu pouvais voir l'intérieur de la plaque, tu verrais que ces rangées sont perpendiculaires aux grandes rangées d'alimentation placées au bord. Les trous sont groupés par cinq et ne sont pas reliés au travers de l'espace au milieu de la plaque.

Construction de ton premier circuit

Te voilà maintenant prêt à réaliser ton circuit à LED utilisant une résistance de 220 ohms (Ω) et une LED. Une LED possède un sens (on dit qu'elle est polarisée), cela signifie que tu peux, par erreur, la relier à l'envers dans un circuit. Si tu observes une LED de près, tu verras que ses deux fils ne sont pas de la même longueur. Le fil le plus long doit être relié au +5 V du circuit et le plus court à la masse. La résistance, elle, n'est pas polarisée, ses deux fils sont donc de longueur identique et peu importe lequel est relié aux différentes parties du circuit.

Suis les étapes suivantes pour construire le circuit de la figure 1-16.

1. Relie une extrémité d'un fil de pontage à une des broches GND d'Arduino et l'autre extrémité à une des grandes rangées de la plaque de montage. C'est le fil noir sur la figure 1-16, mais ton fil peut être de n'importe quelle couleur.
2. Insère un fil de la résistance dans une des petites rangées au milieu de la plaque.

Copyright © 2015 Eyrolles.

3. Insère l'autre fil de la résistance dans une autre petite rangée au milieu de la plaque de montage. Ce doit être une rangée différente de celle utilisée pour le premier fil.
4. Insère le fil le plus long de la LED dans un trou d'une des rangées utilisées par la résistance. Elles sont maintenant reliées à la même partie métallique, le courant électrique pourra donc circuler au travers de la résistance puis dans la rangée de la plaque de montage et dans la LED.
5. Insère le fil le plus court de la LED dans un trou de la grande rangée où le fil de pontage GND est relié (directement ou par l'intermédiaire d'une rangée courte et d'un fil de pontage).
6. Utilise un deuxième fil de pontage pour relier la broche 13 à la petite rangée sur laquelle est relié le fil de la résistance non relié à la LED.

Ne relie jamais la broche +5 V et une broche de masse (GND) ensemble sans intercaler une résistance ou une LED entre elles : cela créerait un court-circuit et pourrait endommager ton Arduino Uno. Si ça arrive par erreur, ton ordinateur s'apercevra probablement que quelque chose d'anormal se passe dans ta carte Arduino et coupera son alimentation électrique vers la carte. Si cela arrive, déconnecte simplement la carte Arduino de l'ordinateur puis reconnecte-la.

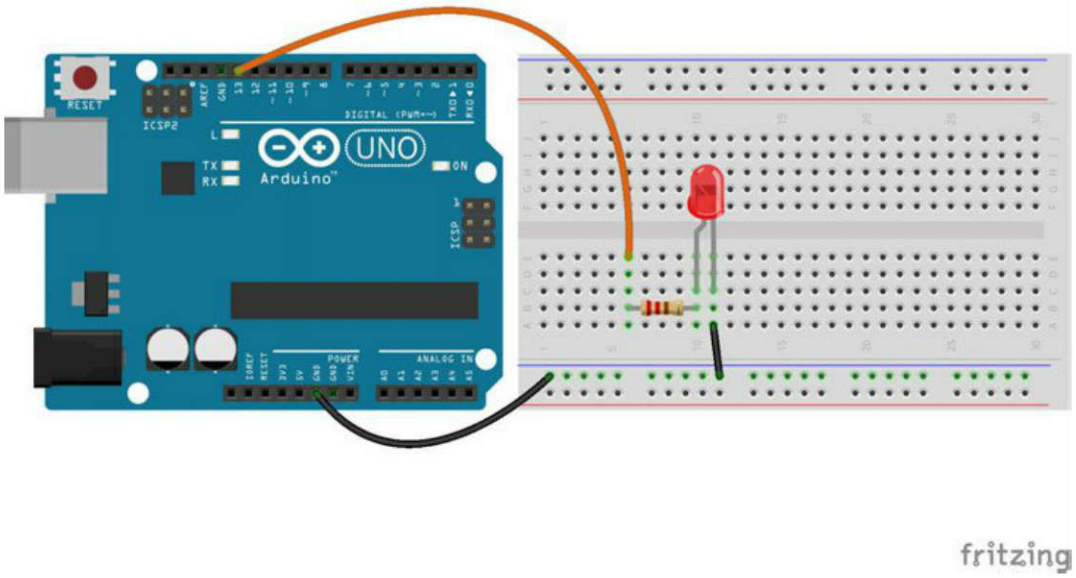


FIGURE 1-16 Le circuit à LED sur la plaque de montage.

Copyright © 2015 Eyrolles.

La LED sur la plaque de montage devrait maintenant clignoter exactement comme le faisait la LED sur la carte Arduino. Félicitations ! Tu as construit ton premier circuit Arduino ! Ton programme contrôle l'électricité et le clignotement de la lumière. Mais ce n'est que le début de tes aventures… Beaucoup d'autres projets sympas t'attendent !

Rends-toi sur le site web d'accompagnement www.editions-eyrolles.com/go/arduino pour voir une vidéo (en anglais) montrant comment construire ce circuit.

DÉFI

Change le numéro de la broche à laquelle est relié ton circuit LED. Tu peux utiliser n'importe laquelle des broches 2 à 13 de la zone Digital, montrée sur la figure 1-17. Les autres broches ont une fonction particulière que tu étudieras dans la prochaine aventure.

Souviens-toi que tu dois faire une modification dans ton code (et téléverser le programme modifié dans ta carte Arduino) et que tu dois aussi modifier ton circuit.

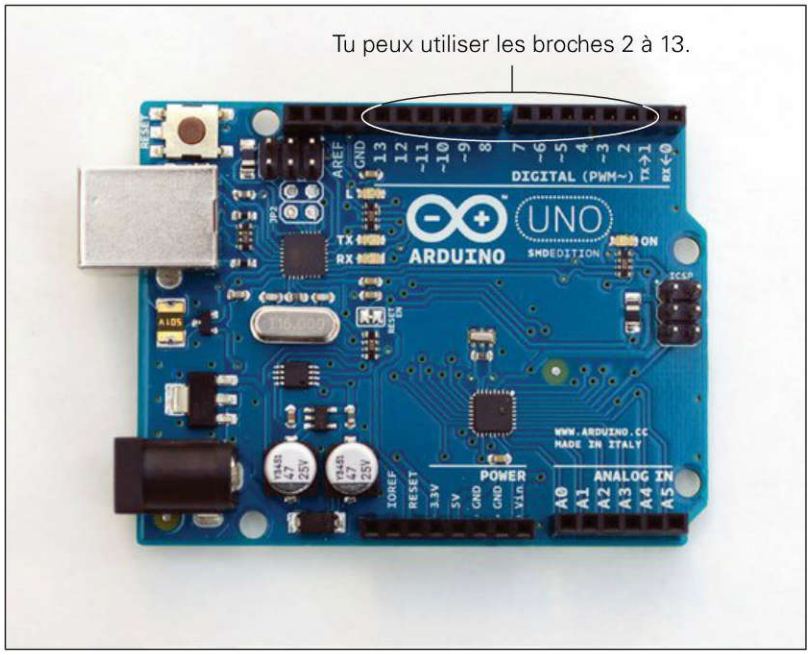


FIGURE 1-17 Les broches numériques de la carte Arduino. Les broches 0 et 1 sont des broches spéciales que tu étudieras plus tard.

Copyright © 2015 Eyrolles.

Plus d'aventures en Arduino

Félicitations ! Tu as été très fort. Tu trouves peut-être que ça représente beaucoup de travail pour simplement faire clignoter une LED, mais en réalité cela signifie que tu es maintenant prêt pour toutes sortes d'aventures.

Regarde les projets que d'autres ont réalisés pour te faire une idée de ce qui est possible :

- `http://makezine.com/category/electronics/arduino/`
- `www.creativeapplications.net/tag/arduino/`

Rappel des commandes Arduino de cette aventure

Commande	Description
`setup()`	Fonction qui ne s'exécute qu'une fois au premier démarrage de l'Arduino Uno. Voir aussi `http://arduino.cc/en/Reference/Setup`.
`loop()`	Fonction qui s'exécute de façon répétitive après que la fonction `setup()` ait terminé et jusqu'à ce que Arduino soit éteinte. Voir aussi `http://arduino.cc/en/Reference/Loop`.
`pinMode()`	Définit si la broche dont le numéro est fourni en argument est utilisée en sortie d'électricité ou en entrée. Voir aussi `http://arduino.cc/en/Reference/PinMode`.
`OUTPUT`	Mot-clé, second argument de la fonction `pinMode()` qui indique que la broche va délivrer une tension électrique. Voir aussi `http://arduino.cc/en/Reference//Constants`.
`digitalWrite()`	Provoque la sortie d'une tension électrique ou sa disparition sur la broche indiquée. Voir aussi `http://arduino.cc/en/Reference/DigitalWrite`.
`HIGH`	Mot-clé utilisé pour délivrer une tension électrique dans la fonction `digitalWrite()`. Voir aussi `http://arduino.cc/en/Reference/Constants`.
`LOW`	Mot-clé utilisé pour interrompre la sortie d'une tension dans `digitalWrite()`. Voir aussi `http://arduino.cc/en/Reference/Constants`.
`delay()`	Provoque une pause de l'Arduino Uno pendant le nombre de millisecondes spécifié en argument. Voir aussi `http://arduino.cc/en/Reference/Delay`.

Copyright © 2015 Eyrolles.

Étape franchie : Tu sais faire les bons branchements maintenant !

Dans la prochaine aventure

Tu commenceras à ajouter de l'interactivité dans ton code Arduino et à contrôler des LED en utilisant un bouton rotatif !

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Aventure 2
Utilise des capteurs

TU AS MAINTENANT installé et fait fonctionner ton logiciel, et tu y as téléchargé ton premier sketch pour t'assurer que tout était correctement installé. (Si tu ne l'as pas fait, il serait bon que tu retournes à l'aventure 1 et que tu le fasses maintenant !) C'est super de pouvoir dire à Arduino de contrôler quelque chose, comme faire clignoter une LED ; mais c'est encore plus amusant quand les projets deviennent interactifs. Il y a deux choses qui doivent arriver pour qu'un projet le devienne : tout d'abord, Arduino doit savoir ce qui se passe dans son environnement ; ensuite, son programme doit exécuter une action basée sur les informations recueillies.

Tu vas parcourir un long chemin dans cette aventure ! Tu vas commencer par contrôler plusieurs LED, puis tu vas afficher des messages envoyés à ton ordinateur depuis la carte Arduino. Ensuite, tu vas lire des informations fournies par un capteur et les envoyer à ton ordinateur pour qu'il les affiche. Pour terminer, tu vas mettre à profit toutes tes connaissances pour réaliser un afficheur de messages effrayants, que tu choisiras à l'aide d'un bouton rotatif. Parfait pour accueillir ou éloigner un visiteur en fonction de ton humeur !

Ce dont tu as besoin

Tu vas d'abord apprendre à ajouter plusieurs LED à ton circuit puis à utiliser un capteur appelé potentiomètre. La liste suivante indique ce dont tu as besoin, la figure 2-1

Copyright © 2015 Eyrolles.

montre les composants électroniques. Souviens-toi que l'annexe B contient des suggestions d'endroits où tu peux acheter ces éléments :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 3 LED (une verte, une jaune et une rouge) ;
- un potentiomètre de 10 kΩ ;
- 4 fils de pontage.

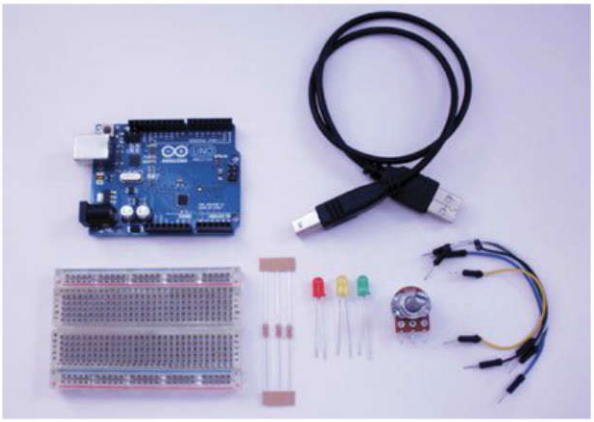


FIGURE 2-1 Les composants électroniques dont tu as besoin pour la première partie de cette aventure.

Ajout de plusieurs LED

Dans la première aventure, tu as réalisé un circuit sur une plaque de montage pour contrôler une seule LED avec ta carte Arduino. Une LED c'est bien, plusieurs c'est encore mieux ! De quoi as-tu besoin pour ajouter plusieurs LED ?

Tu as d'abord besoin de plusieurs LED : cette aventure en utilise trois. Tu as également besoin de trois résistances de 220 Ω, car chaque LED a besoin d'une résistance pour limiter le courant qui la traverse. Les résistances permettent de contrôler le courant électrique dans un circuit afin de protéger les LED contre la destruction qui résulterait d'un courant trop fort. Tu peux lire plus d'informations au sujet du courant électrique et des résistances dans l'aventure 1.

Commence par construire le circuit présenté sur la figure 2-2.

1. Introduis un fil d'une résistance dans une rangée courte de la moitié haute de la plaque de montage, près du bord gauche de la plaque. Insère l'autre fil de la

Copyright © 2015 Eyrolles.

résistance dans la rangée courte de la moitié basse de la plaque, dans le prolongement de la rangée où tu as inséré le premier fil de la résistance. Les liaisons électriques des rangées courtes sont interrompues par l'espace au milieu qui les sépare. Ainsi, les deux fils de la résistance que tu viens de brancher ne sont pas reliés électriquement et possèdent chacun leur propre rangée de contacts.

2. Fais de même avec les deux autres résistances. Place la deuxième au centre de la plaque et la troisième vers la droite. Chaque résistance doit chevaucher l'espace de séparation du milieu de la plaque : avoir un fil inséré dans une rangée au-dessus de cette séparation et l'autre fil en dessous.

3. Ajoute maintenant les LED. Le fil le plus long d'une LED doit être relié à une résistance et l'autre fil à la masse (GND). Insère le fil long de chaque LED dans la rangée qui se situe au-dessous d'une résistance. Place la LED verte à gauche, la jaune au centre et la rouge à droite de la plaque.

4. Insère le fil court de chaque LED dans une des grandes rangées de la longueur inférieure de la plaque. Si ta plaque de montage comporte une ligne bleue ou noire ou le signe -, insère les trois fils courts des LED dans cette rangée. Si elle ne possède pas de marquage, assure-toi simplement que les trois fils sont bien insérés dans la même rangée.

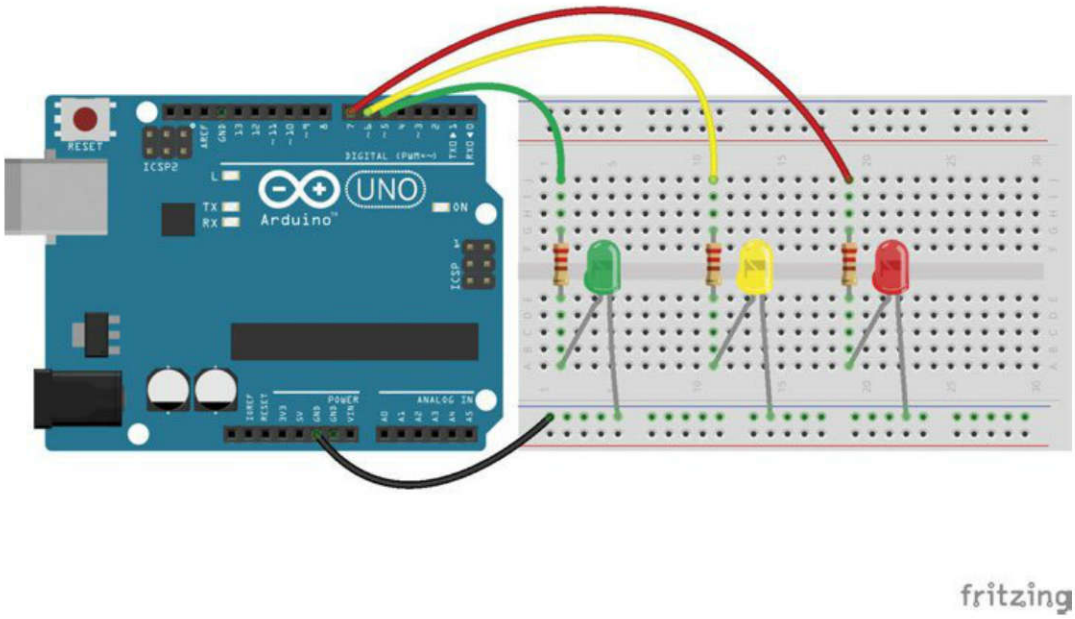


FIGURE 2-2 Construction du circuit contrôlant plusieurs LED.

5. Ton circuit est maintenant réalisé sur ta plaque de montage : il ne reste plus qu'à le relier à ta carte Arduino Uno. Utilise un fil de pontage pour relier une broche GND (il y en a trois sur ta carte Arduino Uno, mais tu peux choisir celle que tu veux, car elles sont identiques) à la rangée à laquelle tu as relié les fils courts des trois LED.

Copyright © 2015 Eyrolles.

6. En utilisant trois fils de pontage supplémentaires, relie un fil à la broche 5 de la carte Arduino (non pas A5, mais celle marquée 5 dans la zone Digital) et au fil du haut de la résistance de gauche sur la plaque, qui est reliée à la LED verte. Utilise le second fil de pontage pour relier la broche 6 à la résistance du milieu, qui est reliée à la LED jaune. Enfin, avec le troisième fil de pontage, relie la broche 7 à la dernière résistance qui est reliée à la LED rouge.

Tu as terminé ? Tu es maintenant prêt à écrire le code. Il va ressembler au code du sketch Blink décrit dans l'aventure 1. Tu vas utiliser ton code pour contrôler quand chaque LED s'allume ou s'éteint. La grande différence, ici, est que tu vas utiliser des variables pour identifier les LED et que tu vas en contrôler trois et non plus une seule. Tu en apprendras davantage au sujet des variables dans la section suivante.

Commence par lancer l'IDE Arduino, il va ouvrir une fenêtre d'un nouveau sketch dès qu'il aura démarré. Tu peux aussi utiliser le menu Fichier > Nouveau ou cliquer sur le bouton Nouveau pour créer un sketch vide. Saisis les lignes de code ci-dessous exactement comme elles sont écrites. Les espaces et les retraits (indentations) ne sont pas importants, ils permettent seulement une meilleure lisibilité du code. N'oublie surtout pas un « ; », sinon ton code ne fonctionnera pas. N'oublie pas aussi de sauvegarder ton sketch en utilisant le menu Fichier > Enregistrer ou en cliquant sur le bouton Enregistrer.

```
// The pins
int greenLED = 5;
int yellowLED = 6;
int redLED = 7;

void setup() {
  // set to output to LED pins
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);
}

void loop() {
  // turn on all LEDs
  digitalWrite(greenLED, HIGH);
  digitalWrite(yellowLED, HIGH);
  digitalWrite(redLED, HIGH);

    // wait 1 second
    delay(1000);

    // turn off all LEDs
    digitalWrite(greenLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(redLED, LOW);
```

Copyright © 2015 Eyrolles.

```
    // wait 1 second
    delay(1000);
}
```

Le code ci-dessus est l'original en anglais, en voici une version en français. Tu peux utiliser celui que tu préfères :

```
// Les broches
int LEDverte = 5;
int LEDjaune = 6;
int LEDrouge = 7;

void setup() {
  // paramétrer les broches des LED en mode sortie (OUTPUT)
  pinMode(LEDverte, OUTPUT);
  pinMode(LEDjaune, OUTPUT);
  pinMode(LEDrouge, OUTPUT);
}

void loop() {
  // allumer toutes les LED
  digitalWrite(LEDverte, HIGH);
  digitalWrite(LEDjaune, HIGH);
  digitalWrite(LEDrouge, HIGH);

    // attendre 1 seconde
    delay(1000);

    // éteindre toutes les LED
    digitalWrite(LEDverte, LOW);
    digitalWrite(LEDjaune, LOW);
    digitalWrite(LEDrouge, LOW);

    // attendre 1 seconde
    delay(1000);
}
```

Entrer le code manuellement est une bonne façon de se familiariser avec les nouvelles instructions que tu apprends, mais cela peut être frustrant quand il s'agit d'un sketch très long ou si tu as commis une erreur de frappe que tu ne trouves pas. Si besoin, tous les sketches sont disponibles en téléchargement sur le site d'accompagnement : www.widley.com/go/adventuresinarduino.

Copyright © 2015 Eyrolles.

Lorsque tu auras terminé la saisie du code de ton sketch et que tu l'auras enregistré, tu seras prêt pour le télécharger dans ta carte Arduino. Choisis le type de carte Arduino et le port à partir des menus (si tu as oublié comment faire, retourne dans l'aventure 1). Clique ensuite sur le bouton Téléverser et vérifie les messages qui s'affichent dans la fenêtre du bas de l'IDE Arduino.

Si aucun message d'erreur (de couleur orange) n'apparaît, alors les trois LED devraient clignoter. C'est beaucoup plus excitant qu'une seule LED clignotante, n'est-ce pas ?

Si tu vois des erreurs (indiquées en anglais en couleur orange), c'est que tu as probablement fait une faute de frappe. Vérifie ton code avec attention et assure-toi qu'il correspond bien à ce que tu as saisi au clavier. S'il y a une erreur en provenance d'`avrdude`, vérifie que ta carte Arduino Uno est bien reliée à ton ordinateur et que tu as choisi le bon port de communication et le bon type de carte. Si tu as encore des problèmes, reviens à l'aventure 1, pour vérifier que ton logiciel est installé correctement.

EXPLORATION DU CODE

Maintenant que tu as vu ce que fait le programme, il est temps de comprendre comment il le fait ! Le sketch peut être divisé en trois parties : les instructions précédant les fonctions, la fonction `setup()` et la fonction `loop()`.

Voici les premières lignes de code précédant les fonctions :

```
// The pins
int greenLED = 5;
int yellowLED = 6;
int redLED = 7;
```

Ou, pour le code francisé :

```
// Les broches
int LEDverte = 5;
int LEDjaune = 6;
int LEDrouge = 7;
```

La première ligne commençant par `//` est un commentaire expliquant ce que les lignes suivantes signifient. Les trois lignes suivantes sont des **variables** auxquelles on affecte les numéros des broches qui sont reliées aux LED. Une variable est un élément du code portant un nom et contenant une valeur. Dans ce sketch, le nom donné permet de se rappeler plus facilement quelle LED est reliée à chaque broche. Le nom `LEDverte` est plus compréhensible que le numéro `5`, n'est-ce pas ?

Copyright © 2015 Eyrolles.

Une variable est un élément du code qui contient une valeur qui peut être modifiée. Par exemple, la variable LED verte contient le numéro 5.

En tête de chaque nom de variable se trouve le mot clé `int`. Cela indique quel type de donnée est stocké dans la variable. Un `int` étant un nombre entier court, les variables précédentes ne peuvent donc mémoriser que des valeurs entières.

Une variable permet de donner un nom à quelque chose, ce qui est très pratique, et le fait de pouvoir changer sa valeur sans modifier son nom est une possibilité très puissante. Cette caractéristique des variables n'est pas utilisée ici, tu le verras dans la section suivante.

Voici quelques règles dont tu dois te rappeler quand tu définis le nom d'une nouvelle variable. Tout d'abord, il ne peut pas contenir d'espaces, tu peux en revanche utiliser un tiret normal ou bas (« LED_verte » est correct, mais pas « LED verte »). Tu ne peux pas commencer le nom d'une variable par autre chose qu'une lettre (« LED3 » est correct, mais pas « 3LED »). Tu ne peux pas utiliser de caractères accentués. Enfin, mais ce n'est pas une obligation, les variables commencent en général par une lettre minuscule (« couleurVerte » plutôt que « CouleurVerte »).

La partie suivante du sketch est la fonction `setup()`. Cette fonction est celle que ton Arduino exécute lorsqu'il démarre. Cette partie n'est exécutée qu'une seule fois, cette fonction est donc prévue pour les commandes qui ne doivent être appliquées qu'une fois. L'action la plus commune réalisée dans la fonction `setup()` est le paramétrage du mode de fonctionnement des broches à l'aide de la commande `pinMode()`. Cette commande détermine si une broche est utilisée en sortie, fournissant une tension électrique, par exemple à une LED, ou en entrée, lorsqu'elle lira un signal en provenance d'un capteur.

Ce circuit n'utilise pas de capteur, mais uniquement des LED, c'est pourquoi `pinMode()` paramètre chaque broche en mode `OUTPUT` (sortie). Puisque des variables sont utilisées pour représenter les numéros des broches, leurs noms (`greenLED`, `yellowLED` et `redLED`) peuvent être employés au lieu de `5`, `6` et `7`.

Copyright © 2015 Eyrolles.

```
void setup() {
// set to output to LED pins
pinMode(greenLED, OUTPUT);
pinMode(yellowLED, OUTPUT);
pinMode(redLED, OUTPUT);
}
```

Ou, pour la version francisée :

```
void setup() {
// paramétrer les broches des LED en
// mode sortie (OUTPUT)
pinMode(LEDverte, OUTPUT);
pinMode(LEDjaune, OUTPUT);
pinMode(LEDrouge, OUTPUT);
}
```

La dernière partie du code est la fonction `loop()`. Cette fonction est exécutée de façon répétitive tant que la carte Arduino est alimentée électriquement.

La fonction `loop()` utilise deux autres fonctions : `digitalWrite()` et `delay()`. La fonction `digitalWrite()` permet d'allumer ou d'éteindre une LED. Son premier argument détermine quelle LED est concernée, le second argument définit ce qui est réalisé : allumer la LED si l'argument est `HIGH` (HAUT) ou l'éteindre si l'argument est `LOW` (BAS).

Pour résumer ce que fait la fonction `loop()` : elle allume les LED pendant une seconde afin que tu puisses constater leur état, puis elle les éteint et attend à nouveau une seconde pour que tu puisses voir qu'elles sont éteintes.

```
void loop() {
  // turn on all LEDs
  digitalWrite(greenLED, HIGH);
  digitalWrite(yellowLED, HIGH);
  digitalWrite(redLED, HIGH);

  // wait 1 second
  delay(1000);

  // turn off all LEDs
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, LOW);
  digitalWrite(redLED, LOW);

  // wait 1 second
  delay(1000);
}
```

Copyright © 2015 Eyrolles.

Ou, pour la version francisée :

```
void loop() {
  // allumer toutes les LED
  digitalWrite(LEDverte, HIGH);
  digitalWrite(LEDjaune, HIGH);
  digitalWrite(LEDrouge, HIGH);

  // attendre 1 seconde
  delay(1000);

  // éteindre toutes les LED
  digitalWrite(LEDverte, LOW);
  digitalWrite(LEDjaune, LOW);
  digitalWrite(LEDrouge, LOW);

  // attendre 1 seconde
  delay(1000);
}
```

DÉFI

Maintenant que tu as trois LED clignotant simultanément, fais-les clignoter une par une en séquence. Tu n'as pas à modifier le circuit ; tu dois uniquement changer le programme. N'oublie pas ensuite de télécharger ton code modifié dans ta carte Arduino Uno.

Quelques recommandations :

- tu ne dois pas écrire de nouvelles fonctions `digitalWrite()`, tu dois juste déplacer celles qui existent déjà ;
- tu dois utiliser des appels supplémentaires à la fonction `delay()`.

Copyright © 2015 Eyrolles.

Envoi de messages à ton ordinateur

Une fois que tu as téléchargé ton code dans ta carte Arduino Uno, celle-ci ne communique plus avec ton ordinateur, car elle n'en a plus besoin pour fonctionner. Le seul rôle de l'ordinateur est alors de lui fournir de l'alimentation électrique au travers du câble USB. Tu pourrais même débrancher ce câble de ton Arduino et utiliser une pile (tu trouveras des informations sur l'alimentation d'une carte Arduino Uno dans l'aventure 5). Ton ordinateur peut cependant être utile après avoir téléchargé le code de ton sketch dans ton Arduino. Par exemple, une carte Arduino n'a pas d'écran alors qu'un ordinateur en possède un. Ainsi, si tu **débogues** (***debug***, en anglais) ton programme, c'est-à-dire si tu le mets au point ou si tu veux lire des messages t'indiquant ce que fait la carte, l'ordinateur et le câble USB seront des outils précieux.

Déboguer (***debug***, en anglais) consiste à déterminer la cause des erreurs dans ton programme et à les corriger.

Ta carte Arduino Uno peut communiquer avec ton ordinateur en utilisant une transmission en série par le câble USB. Cela signifie simplement qu'un seul bit de donnée est envoyé à la fois. Tu peux comparer cela à une seule lettre d'un mot envoyée à la fois, comme pour épeler un mot entier. Pour transmettre des données depuis Arduino vers ton ordinateur, tu as seulement trois fonctions à connaître.

Une **transmission série** (***serial communication***, en anglais) est une technique que deux appareils tels qu'un ordinateur et une carte Arduino utilisent pour envoyer et recevoir des messages. Un seul élément du message (bit) est envoyé à la fois.

La première fonction est une de celles qui doivent être appelées une seule fois dans la fonction `setup()` :

```
Serial.begin(9600) ;
```

Cette fonction ne reçoit qu'un argument : la vitesse à laquelle la carte Arduino Uno transmet les données ou les reçoit. Il est important que cette valeur soit la même que celle utilisée par l'ordinateur (tu verras comment par la suite). Si l'ordinateur et la carte envoient ou reçoivent des données à des vitesses différentes, ils seront incapables de se comprendre. À moins que ne soit spécifiée une autre valeur, 9600 est une valeur correcte.

Copyright © 2015 Eyrolles.

Souviens-toi que tu peux toujours chercher des termes dans le glossaire à la fin de ce livre.

Les deux autres fonctions sont :

```
Serial.print(" Ton message se trouve ici ") ;
Serial.println(" Ton message se trouve ici ") ;
```

Ce sont les fonctions qui envoient des messages de la carte Arduino Uno à l'ordinateur. La première, `Serial.print()`, ne transmet pas de caractère **retour à la ligne** (***newline***, en anglais) à la fin du message. En d'autres termes, le curseur n'est pas positionné en début de la ligne suivante après affichage du message. La fonction `Serial.println()`, elle, envoie le caractère de retour à la ligne à la fin du message. Tu peux comparer cela à un message se terminant par l'appui de la touche Entrée.

Un caractère **retour à la ligne** (***newline,*** en anglais) provoque la même action que quand on appuie sur la touche Entrée ou Retour sur un clavier.

La meilleure façon de comprendre ceci est de l'essayer. Il n'y a pas besoin de circuit pour cela, tu as seulement besoin de ta carte Arduino Uno reliée à ton ordinateur. Crée un nouveau sketch comportant le code suivant et télécharge-le dans ta carte Arduino :

```
void setup() {
  // to start serial communication
  // the argument needs to match
  // the rate you choose in the
  // Serial Monitor
  Serial.begin(9600);
  Serial.println("Hello, this is from setup");

  // a delay so that messages aren't too quick to read
  delay(3000);
}

void loop() {
  // printing a message and then waiting a second
  Serial.print("This is from loop, with a print. ");
```

Copyright © 2015 Eyrolles.

```
  delay(1000);
  Serial.println("And this is from loop with a println.");
  delay(1000);

  int myVariable = 27;
  Serial.print("And this is printing a variable : ");
  Serial.println(myVariable);
  delay(1000);
}
```

Le code ci-dessus est l'original en anglais, en voici une version en français. Tu peux utiliser indifféremment l'un ou l'autre :

```
void setup() {
  // Pour démarrer une communication en série,
  // l'argument doit correspondre à la
  // vitesse de transmission du moniteur série
  Serial.begin(9600);
  Serial.println("Hello, cela vient de la fonction setup");

  // Délai ajouté pour éviter un temps de lecture trop court
  delay(3000);
}
void loop() {
  // Envoi du message et attente d'une seconde
  Serial.print("Envoi depuis la fonction loop par ↩
  Serial.print. ");
  delay(1000);
  Serial.println("Et ceci de la fonction loop par println.");
  delay(1000);

  int maVariable = 27;
  Serial.print("Et ceci est le contenu de ta variable : ");
  Serial.println(maVariable);
  delay(1000);
}
```

Tu as sans doute remarqué que les messages que tu as envoyés par `Serial.print()` et `Serial.println()` sont placés entre des guillemets (`""`). C'est de cette façon que doit être écrit du texte dans un programme pour que l'ordinateur n'interprète pas ce texte comme du code. Tu peux écrire le contenu d'une variable en remplaçant le message et ses guillemets (`""`) par le nom d'une variable, par exemple `maVariable` dans ce sketch.

Copyright © 2015 Eyrolles.

Après que le sketch est téléchargé dans ta carte Arduino, ouvre le moniteur série en cliquant le dernier bouton à droite de l'IDE, celui que je n'ai pas décrit dans l'aventure 1. C'est le bouton isolé à droite sur la figure 2-3. Après avoir cliqué dessus, une fenêtre s'ouvre comme sur la figure 2-4. Tu pourras constater que le nombre 9600 apparaît en bas à droite de cette fenêtre (si ce n'est pas le cas, tu dois cliquer sur le nombre affiché et choisir 9600). C'est à cette valeur que doit correspondre l'argument de la fonction `Serial.begin()` dans ton code Arduino.

Tu verras alors les messages en provenance de ta carte Arduino Uno apparaître dans ton moniteur série. Le moniteur série est un outil intégré à l'IDE Arduino qui te permet de voir les messages envoyés par ta carte en utilisant la communication série. Tu verras que ça te sera très utile dans toutes tes aventures ! Quand tu auras fini d'observer le moniteur série, tu pourras simplement fermer la fenêtre ; celle-ci se fermera automatiquement si tu télécharges un nouveau sketch dans ta carte. Tu peux aussi rouvrir le moniteur série pour voir d'autres messages envoyés.

FIGURE 2-3 Le bouton du moniteur série.

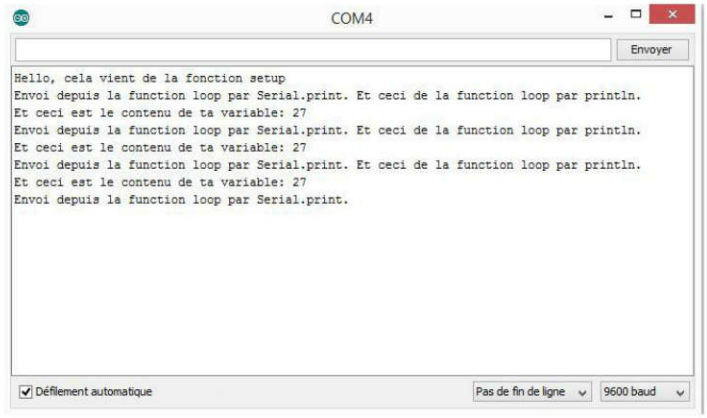


FIGURE 2-4 Le moniteur série de l'IDE Arduino.

Copyright © 2015 Eyrolles.

EXPLORATION DU CODE

Alors que se passe-t-il dans le code ? Il y a des parties avec lesquelles tu es déjà familier : les fonctions `setup()` et `loop()`. Il se passe peu de choses dans la fonction `setup()`. Les premières lignes sont des commentaires expliquant ce qui ce passe. Le port de communication série est démarré et un message est envoyé : `Hello, cela vient de la fonction setup()`. L'Arduino Uno est ensuite mis en pause pendant 3 secondes pour que ce message ne s'affiche pas trop rapidement afin d'en permettre la lecture.

```
void setup() {
  // Pour démarrer une communication en série,
  // l'argument doit correspondre à la
  // vitesse de transmission du moniteur
  //série
  Serial.begin(9600);
  Serial.println("Hello, cela vient de la fonction setup");

  // Délai ajouté pour éviter un temps de lecture trop
  // court
  delay(3000);
}
```

La fonction `loop()` envoie ensuite des messages de différentes façons. En premier, elle utilise `Serial.print()` pour envoyer `Envoi depuis la fonction loop par Serial.print.` Puis elle attend 1 seconde. Puisque ce premier message a été envoyé par `Serial.print()` et non par `Serial.println()`, le message suivant apparaît sur la même ligne à la suite du premier et une deuxième pause d'1 seconde est ensuite effectuée. Puis une variable contenant la valeur numérique `27` est créée. Un appel à la fonction `Serial.print()` provoque l'envoi d'un message suivi de la valeur de la variable à la fin de la ligne. La variable est écrite sans être entourée par des guillemets, c'est donc sa valeur `27` qui est affichée et non pas le texte «maVariable».

```
void loop() {
  // Envoi du message et attente d'une seconde
  Serial.print("Envoi depuis la fonction loop par ↩
  Serial.print. ");
  delay(1000);
  Serial.println("Et ceci de la fonction loop par ↩
  println.");
  delay(1000);
  int maVariable = 27;
  Serial.print("Et ceci est le contenu de ta variable : ");
  Serial.println(maVariable);
  delay(1000);
}
```

Copyright © 2015 Eyrolles.

La fonction `setup()` est exécutée dès que l'Arduino Uno est alimentée, mais celle-ci l'appelle aussi dans deux autres cas après le démarrage. Par exemple, elle contient un bouton *Reset* ou « Réinitialisation » que tu peux utiliser pour la redémarrer afin que la fonction `setup()` soit à nouveau appelée avant de poursuivre avec la fonction `loop()`. De même, chaque fois qu'une nouvelle transmission série est initialisée, la carte redémarre. Ainsi, chaque fois que tu ouvres la fenêtre du moniteur série, la carte redémarre et exécute à nouveau la fonction `setup()`. Le message en cours d'envoi est interrompu et le message contenu dans la fonction `setup()` est envoyé. Ouvre le moniteur série et appuie sur le bouton de réinitialisation de la carte Arduino Uno.

Lecture de données en provenance d'un potentiomètre

Te voilà maintenant prêt à détecter des actions se produisant dans ton environnement et à les utiliser pour prendre des décisions dans ton code. C'est exactement ce pour quoi Arduino a été conçu ! Tu vas très vite pouvoir tourner un bouton rotatif pour changer la LED qui s'illumine.

Tu vas commencer par utiliser un potentiomètre. C'est un nom bien compliqué pour un composant pourtant tout simple ! C'est une résistance dont on peut modifier la valeur par rotation d'un bouton de contrôle ; par exemple, le bouton de réglage du volume de ta radio. Les potentiomètres existent dans différentes formes et tailles. La figure 2-5 montre quelques potentiomètres qui permettent de changer leur valeur par rotation de leur axe. Les deux plus gros peuvent recevoir un bouton rotatif sur leur axe. Le plus petit est parfois appelé potentiomètre ajustable (*trimpot*, en anglais), il possède déjà un petit bouton rotatif que tu peux tourner. Les potentiomètres ajustables, comme celui de la figure 2-5, conviennent parfaitement à la réalisation d'un circuit sur une plaque de montage car ils s'insèrent exactement dans les trous de la plaque. Les potentiomètres plus gros, comme celui de droite, peuvent aussi s'insérer sur une plaque de montage. Celui du milieu est plus pratique pour souder des fils.

Un **potentiomètre** est une résistance possédant un bouton de réglage qui permet de faire varier la résistance du courant électrique.

Copyright © 2015 Eyrolles.

FIGURE 2-5 Différents modèles de potentiomètres.

Tu peux aller sur le site d'accompagnement www.editions-eyrolles.com/go/arduino pour regarder une vidéo montrant plusieurs types de potentiomètres.

Le circuit pour relier le potentiomètre à ton Arduino comporte trois connexions. Tu peux considérer que le potentiomètre possède deux sortes de broches : une paire située aux extrémités et une broche intérieure. La broche intérieure doit être reliée à une broche analogique de ta carte Arduino. Les broches analogiques sont situées dans la partie des broches que tu n'as pas encore utilisées (voir la figure 2-6). Il y en a six au total, chacune est identifiée par la lettre A suivie d'un chiffre (A0, A1, A2, A3, A4 et A5).

FIGURE 2-6 Les broches analogiques d'une carte Arduino Uno.

Copyright © 2015 Eyrolles.

Il est temps que tu construises ton premier circuit utilisant un potentiomètre ! Suis les étapes suivantes pour réaliser le circuit de la figure 2-7.

1. Utilise un fil de pontage pour relier le +5 V de l'Arduino Uno à l'une des grandes rangées situées en bas de ta plaque de montage. Si la plaque comporte un marquage d'une ligne rouge, relie le +5 V à cette rangée, sinon, choisis l'une ou l'autre des deux rangées du bas de la plaque.
2. Utilise un autre fil de pontage pour relier la masse GND de la carte à l'autre rangée du bas de la plaque.
3. Insère les broches du potentiomètre dans des rangées courtes situées au milieu de la plaque.
4. Utilise un fil de pontage pour relier une des broches extérieures du potentiomètre à la rangée utilisée pour la masse GND reliée à la carte Arduino Uno.
5. Utilise un autre fil de pontage pour relier l'autre broche extérieure du potentiomètre à la rangée utilisée pour relier le +5 V à la carte.
6. Relie la broche du milieu du potentiomètre à la broche A0 de la carte.

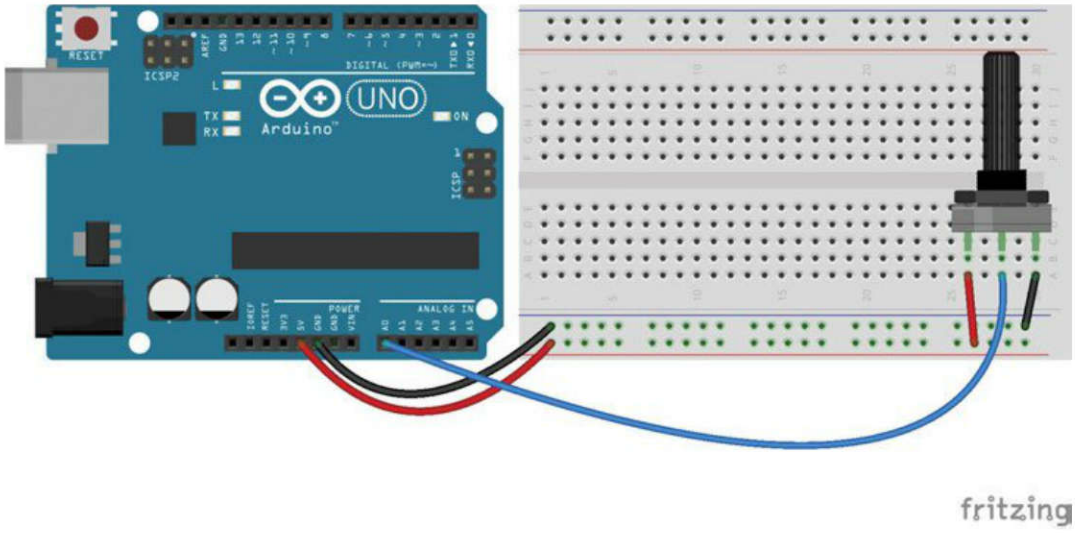


FIGURE 2-7 Le circuit de connexion du potentiomètre.

Tu peux lire la valeur en provenance du potentiomètre, qui correspond à la position de son axe ou de son bouton rotatif, et l'afficher sur le moniteur série. Pour cela, utilise le menu Fichier > Exemples > 01 Basics et ouvre le sketch AnalogReadSerial. Ensuite, téléverse-le dans ta carte Arduino Uno (souviens-toi que tu dois choisir le type de carte et le port comme tu l'as déjà fait dans l'aventure 1 et, précédemment, dans cette aventure).

Quand tu as terminé, clique sur le bouton Moniteur série pour ouvrir la fenêtre du moniteur. Tourne alors le potentiomètre à fond à gauche, puis à fond à droite. Tu

Copyright © 2015 Eyrolles.

devrais voir des nombres s'afficher dans le moniteur série, variant de 0 pour une position du potentiomètre et 1023 pour la position opposée : ce sont les valeurs minimum et maximum qu'Arduino peut lire par une broche d'entrée analogique. Quand la valeur lue est 0, la broche reçoit une tension nulle (0 V), celle de la masse ou GND. Et quand elle est de 1023, cela signifie que la broche reçoit une tension de 5 V. Toute valeur intermédiaire signifie que l'entrée analogique reçoit un niveau compris entre la masse GND et 5 V. 5 V, c'est la tension maximale que peut délivrer et lire une Arduino Uno et 0 V est la minimale. Ainsi, ce circuit permet de déterminer si le potentiomètre est tourné à fond à gauche ou à fond à droite en comparant la tension délivrée par le potentiomètre à la valeur maximale, à la valeur minimale ou à une valeur intermédiaire.

EXPLORATION DU CODE

Que se passe-t-il dans le programme ? Il y a une ligne de code que tu n'as pas encore vue dans le sketch AnalogReadSerial :

```
int sensorValue = analogRead(A0) ;
```

Cette ligne lit la valeur (ou la tension) qui est fournie par le circuit du potentiomètre en utilisant la fonction `analogRead()`. Cette fonction te retourne une valeur comprise entre `0` et `1023`. Tu dois enregistrer cette valeur dans une variable afin de pouvoir la réutiliser plus tard. Une nouvelle variable `sensorValue` (ou `valeurCapteur`) est créée et la valeur que `analogRead()` retourne y est sauvegardée. Cette variable est ensuite envoyée au moniteur série.

DÉFI

Inverse les deux broches extrêmes du potentiomètre afin que celle qui était reliée au +5 V soit maintenant reliée à la masse GND, et que celle qui était reliée à la masse le soit maintenant au +5 V. Tu n'as pas à faire de modification dans le code de l'Arduino Uno.

Que se passe-t-il quand tu modifies le circuit de cette façon par rapport au circuit de la figure 2-7 ?

Tu as dû remarquer qu'Arduino utilise l'orthographe américaine des mots comme « `analog` ». Il est important de s'en rappeler, car une Arduino Uno ne sait pas ce que signifie `analogueRead()`, elle ne connaît que `analogRead()`.

Copyright © 2015 Eyrolles.

Comment prendre des décisions dans ton code ?

Pour réaliser des projets interactifs, tu dois être capable de prendre une donnée en provenance de ton environnement et de faire en sorte que ton Arduino fasse une action dépendante de cette donnée. Cela signifie que tu dois prendre des décisions dans ton programme en fonction des informations délivrées par des capteurs externes. Par exemple, si tu réalises un système d'alarme, tu décideras de ne la faire retentir que si la porte protégée est ouverte, tu dois donc expliquer cela dans ton code.

Les ordinateurs fonctionnent en répondant par oui ou par non à des questions. Ces questions doivent être construites de la façon suivante.

- 3 est-il supérieur à 5 ?
- La valeur 10 est-elle égale à 10 ?
- 4 est-il inférieur ou égal à 8 ?

Une fois traduites en code (afin que l'ordinateur puisse comprendre), ces questions seraient :

```
3 > 5
10 == 10
4 <= 8
```

L'ordinateur ou l'Arduino Uno pourra alors exécuter différentes actions selon que la réponse aux questions sera oui ou non (ou VRAI ou FAUX). On utilise pour cela l'instruction `if` (si, en français). Si la réponse à la question placée entre parenthèses `( )` est oui (ou VRAI), le code placé entre les crochets `{ }` sera exécuté :

```
If (a<b) {
  // Exécuter le code placé ici
}
```

Si la réponse est non (ou FAUX), le code placé entre les crochets { } sera alors ignoré.

Par exemple, le code suivant :

```
If (3<5) {
    serialPrintln (" La réponse au test est VRAI. ") ;
}
serialPrintln (" Ceci est après l'instruction de test. ") ;
```

affichera les messages suivants :

```
La réponse au test est VRAI.
Ceci est après l'instruction de test.
```

Copyright © 2015 Eyrolles.

En revanche, le code suivant :

```
If (3<1) {
    serialPrintln (" la réponse au test est VRAI. ") ;
}
serialPrintln (" Ceci est après l'instruction de test. ") ;
```

affichera uniquement :

```
Ceci est après l'instruction de test.
```

Il est toujours plus facile de comprendre un nouveau concept en le réalisant soi-même et en regardant comment il fonctionne. Dans la section suivante, tu vas utiliser des instructions `if` dans un projet de construction d'un panneau d'affichage de messages.

Construction d'un panneau d'affichage de messages

As-tu déjà vu un studio d'enregistrement, en vrai ou à la télévision ou dans un film ? Il y a habituellement un panneau d'affichage placé à l'extérieur de la pièce qui affiche « ENREGISTREMENT » en s'illuminant afin d'indiquer aux personnes de ne pas entrer quand une session d'enregistrement est en cours.

Tu vas construire ton propre affichage qui indiquera aux autres personnes si elles peuvent entrer, frapper d'abord ou rester à l'extérieur (si tu ne souhaites pas utiliser l'affichage suggéré ici, tu es libre de décider des trois messages que tu voudras afficher). Une LED est placée à côté de chaque message : la LED allumée indique quel message est actif. Tu détermines le message qui doit être actif à l'aide d'un bouton rotatif placé sur le côté du panneau d'affichage, comme sur la figure 2-8.

FIGURE 2-8 Le panneau d'affichage.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

Dans cette aventure, tu vas construire ton premier projet en utilisant ton propre boîtier. Tu as besoin des outils et matériels suivants pour construire puis tester ton circuit sur une plaque de montage, puis pour sa réalisation définitive :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 3 LED (1 verte, 1 jaune, 1 rouge) ;
- 3 résistances de 220 Ω ;
- 1 potentiomètre de 10 kΩ ;
- 8 fils de pontage ;
- du fil de câblage ;
- du ruban isolant électrique ;
- de la soudure ;
- une boîte à chaussures ou toute autre petite boîte ;
- du papier ou de la peinture pour décorer ta boîte ;
- une paire de ciseaux ou un couteau ;
- un fer à souder ;
- une pince coupante ;
- une pince à dénuder ;
- une alimentation USB (optionnelle).

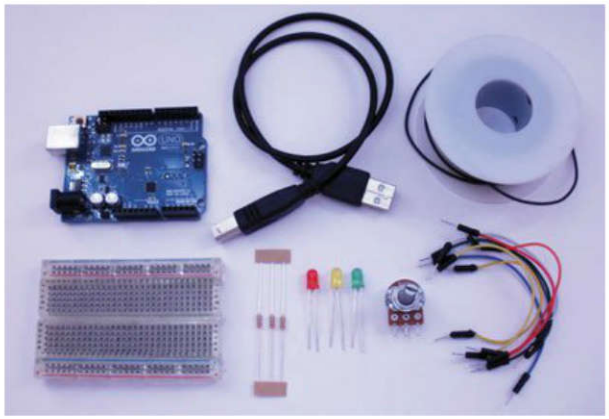


FIGURE 2-9 Les composants électroniques dont tu as besoin pour réaliser ton panneau d'affichage.

Copyright © 2015 Eyrolles.

Compréhension du circuit

Une fois que tu auras soudé et collé les composants, il te sera difficile de tout défaire si tu as commis une erreur. C'est pourquoi je te conseille de commencer toujours par fabriquer un prototype avant de réaliser ton circuit définitif, afin de t'assurer que tout fonctionne correctement. Ainsi, si tu notes une erreur dans la conception de ton projet, tu pourras facilement la corriger avant d'avoir mis en place tous les composants de façon permanente.

La figure 2-10 montre le circuit que tu vas construire pour ton affichage : un circuit utilisant trois LED et un potentiomètre. Les LED seront reliées aux broches 5, 6 et 7, et le potentiomètre à la masse GND, à la broche A0 et à l'alimentation +5 V.

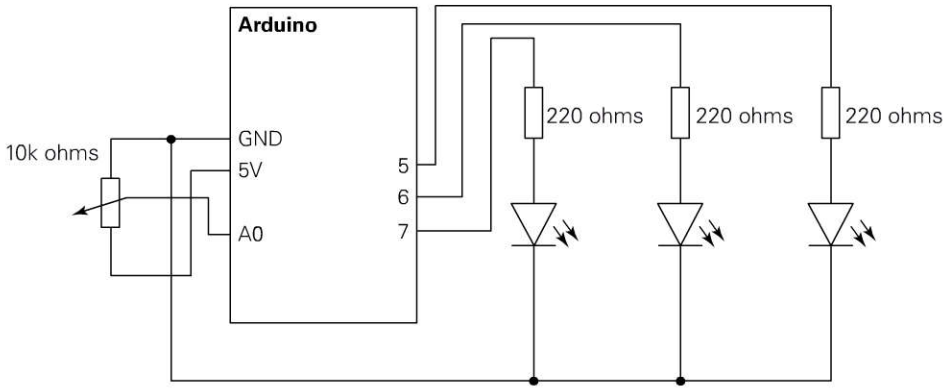


FIGURE 2-10 Le schéma du circuit du panneau d'affichage.

Tu vas maintenant tester le circuit et le code Arduino sur la plaque de montage. Ensuite, tu reconstruiras ton circuit sans utiliser la plaque de montage.

Réalisation du prototype sur ta plaque de montage

Pou réaliser le prototype de ton circuit, tu dois suivre les étapes suivantes.

1. Utilise un fil de pontage pour relier le +5 V de ta carte Arduino Uno à l'une des grandes rangées situées en bas de ta plaque. Si celle-ci possède un marquage identifiant la ligne +5 V (ligne rouge ou +), connecte le +5 V à cette rangée ; sinon tu peux choisir n'importe laquelle des deux rangées.
2. Utilise un autre fil de pontage pour relier la masse GND à l'Arduino Uno en utilisant l'autre grande rangée du bas de la plaque.
3. Insère les broches du potentiomètre dans des rangées courtes, vers le milieu de la plaque de montage.
4. Utilise un fil de pontage pour relier l'une des broches extérieures du potentiomètre à la rangée reliée à la masse.

Copyright © 2015 Eyrolles.

5. Utilise un autre fil de pontage pour relier l'autre broche extérieure du potentiomètre à la rangée reliée au +5 V de l'Arduino Uno.
6. Relie la broche du milieu du potentiomètre à la broche A0 de la carte.
7. Insère l'un des fils de la résistance dans une rangée courte de la moitié haute de ta plaque, vers le bord gauche. Insère l'autre fil dans la rangée courte en face de la précédente, dans la partie basse de la plaque. Les rangées ne sont pas reliées au travers de la séparation ; ainsi, chaque fil de la résistance dans sa propre rangée n'est pas relié à l'autre rangée par une liaison métallique.
8. Répète cela pour la deuxième, puis la troisième résistance. Place l'une au milieu de la plaque et l'autre vers le bord droit. Chaque résistance doit être placée au-dessus de l'espace de séparation des rangées de la plaque et avoir l'un de ses fils inséré dans une rangée de la partie supérieure et l'autre dans la rangée opposée située dans la partie inférieure.
9. Ajoute maintenant les LED. Le fil le plus long de chaque LED doit être relié à sa résistance et le fil le plus court à la masse GND. Insère le fil long de chaque LED dans la même rangée que chaque résistance. Elle devra être placée juste sous la résistance. Place la LED verte du côté gauche de la plaque, la jaune au milieu et la rouge à droite.
10. Insère le fil le plus court de chaque LED dans la rangée longue du bas qui est reliée à la masse GND de l'Arduino Uno.
11. Utilise maintenant trois autres fils de pontage, le premier pour relier la sortie numérique 5 (pas A5, mais 5 dans la section Digital) au fil supérieur de la résistance reliée à la LED verte (placée dans la partie gauche de la plaque). Utilise le deuxième fil de pontage pour relier la broche 6 à la résistance du milieu (reliée à la LED jaune) et le troisième pour relier la broche 7 à la dernière résistance (reliée à la LED rouge).

Quand tu auras terminé, ton circuit prototype devra ressembler à celui de la figure 2-11. Tu ne remarques rien ? Le circuit complet pour le panneau d'affichage est une combinaison des deux circuits avec lesquels tu as déjà travaillé dans cette aventure ! Le potentiomètre est lu par la broche A0 et les trois LED sont contrôlées par les broches 5, 6 et 7.

Écriture du programme

Maintenant, tu as besoin du programme. Lance l'IDE Arduino et entre le code suivant dans la fenêtre d'un nouveau sketch. N'oublie pas de l'enregistrer !

Commence ton sketch en créant des fonctions `setup()` et `loop()` vides.

```
void setup () {
}
void loop () {
}
```

Copyright © 2015 Eyrolles.

Ensuite, ajoute une variable au début du sketch afin de mémoriser l'état de ton potentiomètre.

```
// Les broches
int brochePot = A0;
```

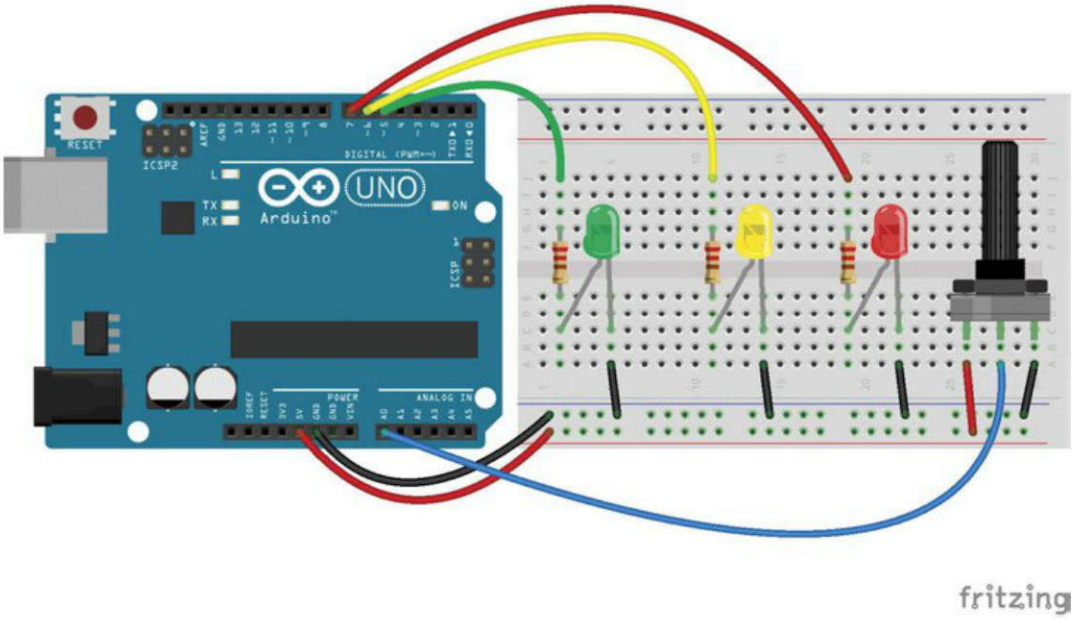


FIGURE 2-11 Le circuit prototype du panneau d'affichage sur la plaque de montage.

Dans la fonction `setup()`, démarre une transmission série afin de pouvoir visualiser des messages sur le moniteur série entre les lignes suivantes, dans les crochets { } de `setup()`.

```
// Démarrage de la transmission série
Serial.begin(9600);
```

La fonction `loop()` contrôle toutes les actions du programme. La valeur en provenance du potentiomètre est lue puis enregistrée dans la variable appelée `valeurPot`. Un message différent est affiché selon la valeur sauvegardée dans cette variable. Le message affiche ce qui doit se passer avec les LED. Saisis les lignes suivantes entre les crochets { } de la fonction `loop()`.

```
int valeurPot = analogRead (brochePot);
// Impression de la valeur du potentiomètre
Serial.print (" Le potentiomètre vaut : ");
Serial.println(valeurPot);

// Si la valeur valeurPot est plus petite que 341
```

Copyright © 2015 Eyrolles.

```
if (valeurPot < 341) {
   Serial.println ("Allumage de la LED verte et ↩
  extinction  des LED rouge et jaune");
}

// Si la valeur valeurPot est plus grande ou égale à 341 et
// inférieure à 682
if (valeurPot >= 341 && valeurPot <682) {
   Serial.println ("Allumage de la LED jaune et ↩
   extinction des LED verte et rouge");
}

// Si la valeur valeurPot est plus grande ou égale à 682
if (valeurPot >= 682) {
   Serial.println ("Allumage de la LED rouge et ↩
   extinction des LED verte et jaune");
}

// Pause pour ralentir l'affichage des messages
delay(50) ;
```

Sauvegarde ton sketch et télécharge-le dans ta carte Arduino Uno. Ouvre ensuite le moniteur série et regarde ce qui se passe quand tu tournes l'axe du potentiomètre. Tu devrais voir la valeur du potentiomètre s'afficher et le comportement des LED – mais tu n'as pas encore programmé les LED. Il est temps de le faire !

Ajoute trois autres variables au début de ton sketch, pour mémoriser les numéros des broches des LED.

```
int LEDVerte = 5;
int LEDJaune = 6;
int LEDRouge = 7;
```

À l'intérieur de la fonction `setup()`, ajoute le code qui définit le mode de chaque broche à l'aide de la fonction `pinMode()`.

```
// Définir le mode OUTPUT (sortie) pour chaque broche des LED
pinMode(LEDVerte, OUTPUT);
pinMode(LEDJaune, OUTPUT);
pinMode(LEDRouge, OUTPUT);
```

Dans la fonction `loop()`, ajoute les appels à la fonction `digitalWrite()` pour allumer et éteindre chaque LED (en caractères gras, dans le code suivant). Tu peux aussi retirer l'appel à la fonction `delay()` à la fin de la fonction `loop()`. Ton sketch complet, devrait ressembler à ceci :

```
// Les broches
int brochePot = A0;
```

Copyright © 2015 Eyrolles.

```
int LEDVerte = 5;
int LEDJaune = 6;
int LEDRouge = 7;

void setup() {
// Définition du mode OUTPUT (sortie) pour chaque broche des
// LED
pinMode(LEDVerte, OUTPUT);
pinMode(LEDJaune, OUTPUT);
pinMode(LEDRouge, OUTPUT);

// Démarrage de la transmission série
Serial.begin(9600);
}

void loop() {
  int valeurPot = analogRead (brochePot);

  // Impression de la valeur du potentiomètre
  Serial.print ("Le potentiomètre vaut :");
     Serial.println(valeurPot);

    // Si la valeur valeurPot est plus petite que 341
    if (valeurPot < 341) {
       Serial.println ("Allumage de la LED verte et ↩
  extinction des LED jaune et rouge");
       // Allumer la LED verte
       digitalWrite(LEDVerte, HIGH);
       // Éteindre les LED jaune et rouge
       digitalWrite(LEDJaune, LOW);
       digitalWrite(LEDRouge, LOW);
  }

  // Si la valeur valeurPot est plus grande ou égale à 341
  // et inférieure à 682
  if (valeurPot >= 341 && valeurPot <682) {
     Serial.println ("Allumage de la LED jaune et ↩
  extinction des LED verte et rouge");
     // Allumer la LED jaune
     digitalWrite(LEDJaune, HIGH);
     // Éteindre les LED verte et rouge
     digitalWrite(LEDVerte, LOW);
     digitalWrite(LEDRouge, LOW);
  }

// Si la valeur valeurPot est plus grande ou égale à 682
  if (valeurPot >= 682) {
    Serial.println ("Allumage de la LED rouge et ↩
```

Copyright © 2015 Eyrolles.

```
  extinction des LED verte et jaune");
     // Allumer la LED rouge
     digitalWrite(LEDRouge, HIGH);
     // Éteindre les LED verte et jaune
     digitalWrite(LEDVerte, LOW);
     digitalWrite(LEDJaune, LOW);
  }
}
```

Téléverse le sketch dans ton circuit réalisé sur la plaque de montage. Si tu ne veux pas écrire à la main tout le programme, tu peux le télécharger (en français ou en anglais) depuis le site d'accompagnement www.editions-eyrolles.com/go/arduino.

La version anglaise du sketch :

```
// The pins
int potPin = A0;
int greenLED = 5;
int yellowLED = 6;
int redLED = 7;

void setup() {
  // set to output to LED pins
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  // start serial
  Serial.begin(9600);
}

void loop() {
  int potValue = analogRead(potPin);
  // print what the pot value is
  Serial.print("Potentiometer is : ");
  Serial.println(potValue);
  // if pot is less than 341
  if(potValue < 341) {
    Serial.println("Turn on green, turn off yellow and red");
    // turn on green LED
    digitalWrite(greenLED, HIGH);
    //turn off yellow and red LEDs
    digitalWrite(yellowLED, LOW);
    digitalWrite(redLED, LOW);
  }

  // if pot more than or equal to 341 and
  // less than 682
  if(potValue >= 341 && potValue < 682) {
```

Copyright © 2015 Eyrolles.

```
    Serial.println("Turn on yellow, turn off green and red");
    // turn on yellow LED
    digitalWrite(yellowLED, HIGH);
    // turn off green and red LEDs
    digitalWrite(greenLED, LOW);
    digitalWrite(redLED, LOW);
  }

  // if pot more than or equal to 682
  if(potValue >= 682) {
    Serial.println("Turn on red, turn off green and yellow.");
    // turn on red LED
    digitalWrite(redLED, HIGH);
    // turn off green and yellow LEDs
    digitalWrite(greenLED, LOW);
    digitalWrite(yellowLED, LOW);
  }
}
```

EXPLORATION DU CODE

Il y a quelque chose dans le code du sketch que tu n'as pas encore rencontré : c'est le caractère spécial `&&`. Ces deux caractères (appelés des esperluettes) sans espace séparateur signifient que les deux parties du code qui le précédent et qui le suivent doivent être VRAI (`TRUE`).

Par exemple :

```
4<6 && 10<20
```

est VRAI, car à la fois `4<6` et `10<20` sont VRAI. En revanche :

```
3>9 && 5<7
```

est FAUX (`FALSE`), car seul `5<7` est VRAI alors que `3>9` est FAUX. Le symbole `&&` est un moyen de combiner des restrictions dans une instruction `if`. Dans ton sketch, il est utilisé pour allumer la LED jaune uniquement si `ValeurPot >= 341` et `valeurPot < 682`.

Tu es prêt ? Alors il est temps d'essayer. Tu devrais être capable de changer la LED qui s'allume en tournant l'axe du potentiomètre. Une seule LED à la fois devrait s'allumer. Tu peux ouvrir le moniteur série de l'IDE Arduino pour t'assurer que des valeurs correctes sont fournies par le potentiomètre.

Copyright © 2015 Eyrolles.

Construction de ton panneau d'affichage

Ce qui rend tout projet vraiment existant, c'est la structure dans laquelle tu loges ton électronique. Elle ne sert pas uniquement à protéger ton circuit et à cacher les parties que tu ne souhaites pas laisser visibles, elle permet aussi d'être créatif. Le programme et le circuit sont une grande partie de la démarche créative dans la réalisation d'un projet Arduino, bien entendu ! Mais c'est dans cette troisième partie que tu peux réellement donner libre cours à ton imagination pour montrer ton projet à ta famille et à tes amis, en retirant ton circuit de la plaque de montage et en le plaçant dans un joli boîtier !

Tu peux choisir n'importe quel matériau que tu désires pour créer ton panneau, mais une boîte à chaussures conviendra parfaitement. Elle peut facilement se découper avec un couteau de bricolage ou des ciseaux et être décorée avec du papier et de la colle ou avec de la peinture pour la personnaliser davantage.

Tu peux regarder une vidéo (en anglais) montrant la réalisation du panneau d'affichage et de son circuit soudé en allant sur le site d'accompagnement www.editions-eyrolles.com/go/arduino.

Perçage des trous pour le potentiomètre et les LED

Avant de décorer ta boîte, tu dois percer quelques trous aux endroits où tu veux que les LED et le bouton du potentiomètre soient situés. Perce cinq trous : trois pour les LED, un pour le potentiomètre et un pour le câble USB. Mesure d'abord le diamètre de la lentille des LED et de l'axe du potentiomètre afin de percer des trous d'un diamètre suffisant pour que ces composants se logent parfaitement dedans. (Si tu perces des trous trop gros, les LED et le potentiomètre tomberont au fond de la boîte !). Pour le câble USB, le trou doit être suffisamment grand pour permettre le passage du connecteur le reliant à ta carte Arduino Uno.

Si tu utilises une boîte à chaussures, je te recommande de faire les trous pour les LED, le potentiomètre et le câble USB dans le fond de la boîte, comme sur la figure 2-12. De cette façon, tu pourras facilement retirer le couvercle pour accéder au circuit électronique et le dissimuler.

Ajout des messages et décoration du panneau

Maintenant, te voilà prêt à transformer une vieille boîte à chaussures en un élégant panneau d'affichage en la peignant ou en la recouvrant de papier. Choisis les messages : tu

Copyright © 2015 Eyrolles.

peux utiliser ceux que j'ai suggérés sur la figure 2-11 ou en inventer d'autres. Tu peux les écrire, les peindre sur la boîte, les imprimer à l'aide de ton ordinateur, ou les découper dans des journaux et les coller à proximité des LED. Cela n'a pas d'importance. Fais comme tu veux, il n'y a pas de limite ! Selon moi, ce sera toujours joli avec des paillettes. Ou bien, pourquoi ne pas utiliser des éléments naturels, comme des plumes ou des fleurs séchées ?

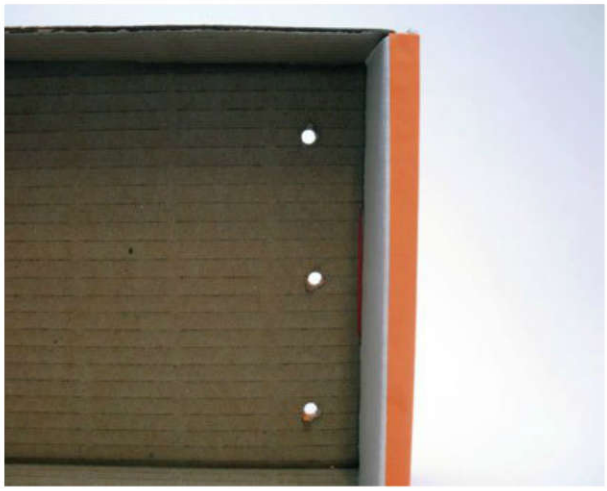


FIGURE 2-12 Perçage des trous pour les LED et pour l'axe du potentiomètre.

Soudage du circuit

Tu sais que ton programme et ton circuit fonctionnent (si tu ne les as pas encore testés, retourne en arrière et fais-le maintenant !). Tu es donc prêt à assembler ton circuit de manière permanente. Les circuits font appel au passage de l'électricité au travers de matériaux conducteurs comme le métal. Cela signifie que tu ne peux pas utiliser de la colle pour relier tes composants, car l'électricité ne peut pas traverser la colle. À la place, tu dois utiliser de la soudure : c'est de la colle qui conduit l'électricité.

La soudure est un métal qui fond à une température plus faible que la plupart des métaux, mais cette température est tout de même très élevée ! Plus élevée que celle pouvant être atteinte par le four de ta cuisine, c'est pourquoi il est important de prendre des précautions en soudant. Sois aussi prudent que si tu manipulais des casseroles ou des poêles très chaudes en cuisinant.

Ne fais de la soudure qu'en présence d'un adulte prêt à t'aider !

Copyright © 2015 Eyrolles.

Il existe de nombreux sites en ligne qui expliquent comment souder, si l'adulte qui t'aidera et toi n'avez jamais effectué de soudures auparavant. Sur YouTube, tu trouveras de nombreuses vidéos d'apprentissage, mais les tutoriels en français, `www.didel.com/Soudure.pdf` et `www.sonelec-musique.com/electronique_bases_tutoriel_soudure.html` sont également excellents pour débuter.

La soudure peut être difficile quand tu réalises qu'il te manque des mains pour maintenir les éléments en place. Tu peux acheter un outil appelé « troisième main » qui t'aidera à maintenir les composants en place. Tu peux aussi utiliser des pastilles adhésives ou de la pâte à fixer (Patafix® de la marque Uhu par exemple) pour immobiliser les composants pendant que tu les soudes.

Quand la peinture et la colle de ta boîte sont sèches, tu peux commencer à réaliser ton circuit. Avant de commencer, tu dois décider de l'endroit où ta carte Arduino Uno sera située dans ta boîte.

Voici comment tu vas réaliser ton circuit (voir la figure 2-13).

1. Connecte une résistance à chacune de tes LED. Entoure l'un des fils de chaque résistance sur le fil le plus long de chaque LED afin qu'il soit maintenu en place. Soude la connexion. Fais de même pour les trois LED.
2. Place les trois LED dans leurs trous dans la boîte. Plie le fil le plus court (celui qui doit être relié à la masse GND) de la LED du haut en direction de la LED du centre. Répète cela pour la LED du milieu, vers la LED du bas.
3. Coupe un morceau de fil de câblage pour relier le fil court de la LED du haut, que tu viens de plier, avec le fil court plié de la LED du centre. De la même façon, coupe un fil de câblage pour relier le fil court plié de la LED du centre au fil court de la LED du bas. Découpe tes fils de câblage environ 2,5 cm plus longs que la distance entre les fils des LED à relier. Utilise une pince à dénuder pour retirer environ 1,25 cm d'isolant à chacune de leurs extrémités.
4. Retire les LED de la boîte. Enroule une extrémité d'un des fils de câblage sur le fil le plus court de la LED du haut. Soude les fils ainsi réunis à l'aide d'un fer à souder.

Copyright © 2015 Eyrolles.

Souviens-toi que tu dois toujours souder en présence d'un adulte. Sois prudent : la pointe d'un fer à souder est brûlante !

5. Enroule l'extrémité de l'autre fil de câblage sur le fil le plus court de la LED du milieu. Soude-les ensemble.
6. Maintenant, tu vas relier le fil de câblage venant du fil le plus court de la LED du haut, au fil le plus court de la LED du centre. Enroule l'extrémité du fil de câblage provenant de la LED du haut autour du fil le plus court de la LED du milieu et soude-les ensemble.
7. De la même manière, relie le fil de câblage provenant de la LED du centre au fil le plus court de la LED du bas.
8. Replace tes LED ainsi connectées dans leurs trous dans la boîte afin de t'assurer qu'elles peuvent toujours y être insérées. Si ce n'est pas le cas, tu peux couper ou dessouder les fils et recommencer.

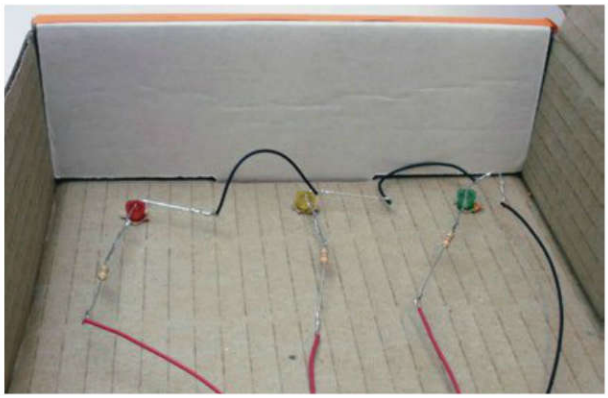


FIGURE 2-13 La partie du circuit qui comporte les LED.

9. Tu vas maintenant couper les fils de câblage qui vont relier les LED à la carte Arduino Uno. Tu dois mesurer et découper trois fils qui partent des résistances connectées aux LED et un fil provenant du fil le plus court de la LED du bas. À nouveau, coupe-les environ 2,5 cm plus longs et dénude-les sur 1,25 cm à chaque extrémité.
10. Soude chaque fil à la LED et aux résistances.

Soude maintenant des fils au potentiomètre (voir la figure 2-14) :

1. Place le potentiomètre dans son trou. Découpe trois fils de câblage pour relier la carte Arduino Uno aux trois broches du potentiomètre.

Copyright © 2015 Eyrolles.

2. Dénude les extrémités de chaque fil sur environ 1,25 cm et soude-le à chaque broche du potentiomètre.
3. Dénude environ 1,25 cm des extrémités libres de chaque fil. Cela te permettra de les insérer directement dans les trous de connexion de la carte Arduino Uno.

FIGURE 2-14 Le potentiomètre soudé.

À ce stade, arrête-toi et inspecte ton travail. Vérifie bien que toutes les parties métalliques du circuit (les fils et les composants) ne se touchent pas accidentellement. Si cela arrivait, des liaisons électriques inopinées se produiraient. Dans ce cas, protège les parties métalliques en cause avec du ruban adhésif pour électricien.

Insertion de l'électronique

Quand la colle et la peinture que tu as utilisées pour décorer ta boîte sont sèches, tu peux terminer ton panneau d'affichage et y insérer ton circuit électronique.

Place les LED dans leurs trous respectifs dans la boîte. Tu peux utiliser un peu de colle pour les maintenir en place si c'est nécessaire.

Le potentiomètre comporte une rondelle et un écrou qui se visse jusqu'à la base de son axe. Retire-les, pousse le potentiomètre dans son trou et visse-le en utilisant la rondelle et l'écrou pour qu'il soit fermement fixé à la boîte. Tu peux placer un bouton rotatif sur l'axe du potentiomètre si tu le souhaites.

Copyright © 2015 Eyrolles.

Insère les fils de ton circuit dans les trous de connexion de ta carte Arduino Uno. Les trois fils reliés aux résistances vont dans les broches 5, 6 et 7. Le fil relié au fil le plus court de la LED du bas est inséré dans une broche GND. Un des fils reliés à une broche externe du potentiomètre est relié à la broche +5 V, et l'autre fil provenant de l'autre broche externe l'est aussi à la broche GND. Le fil restant, qui est relié à la broche du milieu, est inséré dans la broche A0.

Tu dois maintenant décider de la façon dont ton projet va être alimenté. Tu peux bien entendu laisser ta carte Arduino Uno reliée à ton ordinateur, mais cela est peut-être peu pratique. Tu peux aussi utiliser le câble USB en le reliant à un adaptateur secteur comme celui de la figure 2-15, au lieu de ton ordinateur. Les adaptateurs secteur sont souvent fournis avec les téléphones mobiles, tu en as peut-être un déjà disponible. Tout adaptateur qui te permet de connecter un câble USB convient.

FIGURE 2-15 Une alimentation secteur pouvant être utilisée avec une carte Arduino.

Bravo ! Tu viens de réaliser un panneau d'affichage que tu peux installer pour diffuser le message de ton choix. Tu as créé ton premier projet Arduino qui peut se passer de ton ordinateur et fonctionner de manière autonome. Le tien peut paraître très différent de celui de la figure 2-16, mais cela importe peu ! Tu es sur la bonne voie pour devenir un expert en Arduino !

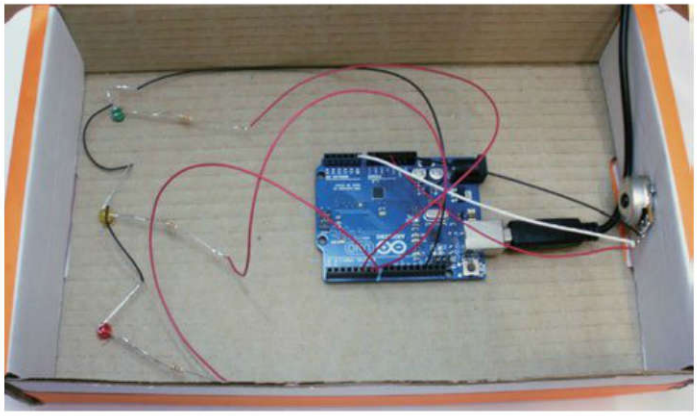


FIGURE 1-16 Le panneau d'affichage terminé.

Copyright © 2015 Eyrolles.

Plus d'aventures avec Arduino

Maintenant que tu peux changer les sorties d'un microcontrôleur Arduino Uno selon la façon dont tu tournes l'axe d'un potentiomètre, que pourrais-tu faire d'autre ? Voici quelques idées de projets :

- changer la vitesse de clignotement d'une LED en tournant le potentiomètre ;
- faire clignoter des LED en séquence et changer la vitesse à l'aide d'un potentiomètre.

Rappel des commandes Arduino de cette aventure

Commande	Description
`int`	Type de donnée qui crée une nouvelle variable entière (un nombre entier). Voir également `http://arduino.cc/en/Reference/Int`.
`Serial.begin()`	Démarre une transmission série permettant l'envoi et la réception de messages. Voir aussi `http://arduino.cc/en/Serial/Begin`.
`Serial.print()`	Envoie un message sans retour à la ligne final. Voir également `http://arduino.cc/en/Reference/Serial/Print`.
`Serial.println()`	Envoie un message avec retour à la ligne à la fin. Voir également `http://arduino.cc/en/Reference/Serial/Print`.
`analogRead()`	Lit la valeur de la tension appliquée à la broche spécifiée et lui affecte une valeur comprise entre 0 (pour le niveau GND) et 1023 (pour le niveau +5 V. Voir également `http://arduino.cc/en/Reference/AnalogRead`.
`if()`	Instruction utilisée pour déterminer si une partie de code doit être exécutée. Voir également `http://arduino.cc/en/Reference/If`.

Copyright © 2015 Eyrolles.

Étape franchie : Tu sais désormais tenir compte de ton environnement et prendre des décisions !

Dans la prochaine aventure

Tu ajouteras du mouvement et tu contrôleras un moteur !

Copyright © 2015 Eyrolles.

Aventure 3

Utilise des servomoteurs

UN MOYEN de rendre tes projets plus dynamiques consiste à y ajouter du mouvement, afin de leur donner de la vie. Ici je vais te montrer comment faire, en utilisant un servomoteur et en ajoutant des interrupteurs à ton circuit.

Dans cette aventure, tu vas commencer par découvrir un nouveau composant, puis tu vas l'utiliser pour réaliser un coffre-fort fantastique, qui ne s'ouvrira que lorsque tu auras réglé tous ses potentiomètres sur leurs positions secrètes et appuyé sur un bouton. Si tu connais la bonne combinaison, ton coffre s'ouvrira automatiquement. Le coffre est construit en carton, il ne résistera donc pas à une attaque brutale, mais il pourra être utilisé pour empêcher tes parents de voir ce qui s'y trouve !

Ce dont tu as besoin

Tu vas utiliser un nouvel **actionneur** : c'est un mot fantaisiste pour désigner un objet qui, à partir d'un signal électrique, effectue une action dans ton environnement. Tu en as déjà utilisé un dans les aventures 1 et 2 : une LED, qui prend l'électricité et la transforme en lumière. Dans cette aventure, tu vas utiliser un moteur qui prend l'électricité et la transforme en mouvement mécanique.

Copyright © 2015 Eyrolles.

Un **actionneur** transforme un signal électrique en une action sur l'environnement, tel qu'un éclairage ou la production d'un son ou d'un mouvement.

À l'opposé de l'actionneur se trouve le **capteur**, et tu vas également en utiliser un nouveau. Dans l'aventure 2, tu as découvert le potentiomètre, qui évalue la rotation d'un axe et la convertit en un signal électrique. Ici, tu vas encore utiliser des potentiomètres, ainsi qu'un bouton-poussoir qui va transformer une poussée en un signal électrique.

Un **capteur** détecte quelque chose dans l'environnement, comme la luminosité, le son ou un mouvement, et le traduit en un signal électrique.

Tu as besoin des éléments suivants. Les composants électroniques sont présentés sur la figure 3-1 :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 4 fils de pontage ;
- un servomoteur ;
- un bouton-poussoir tactile ;
- une résistance de 10 kΩ.

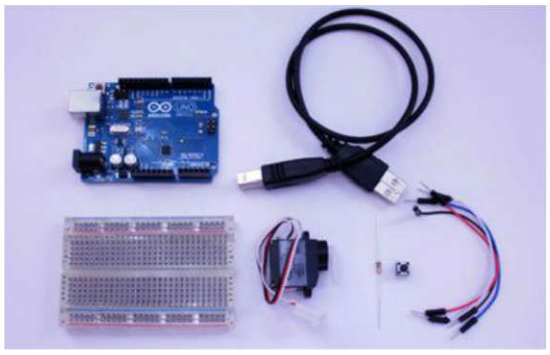


FIGURE 3-1 Les composants électroniques nécessaires à la première partie de l'aventure 3.

Copyright © 2015 Eyrolles.

Description des différents types de moteurs

« Moteur » est le terme général désignant un objet qui utilise un signal électrique et le convertit en un mouvement mécanique. Il existe différents types de moteurs qui te permettent de contrôler des mouvements de différentes façons. Quand tu penses à un moteur, tu imagines probablement un moteur à courant continu : le **courant continu** est le type d'électricité que tu utilises dans tes circuits Arduino.

> Le **courant continu** (*Direct Current* ou **DC** en anglais) est un courant électrique du type de celui utilisé dans les circuits Arduino. C'est un courant électrique identique à celui qui est produit par une pile, à l'opposé du courant alternatif (*Alternating Current* ou **AC** en anglais), qui est présent dans les prises électriques de ta maison.
>
>
>

Tu trouves des petits moteurs électriques à courant continu dans les jouets comme les voitures radiocommandées. Quand un moteur à courant continu est alimenté en électricité, il fait tourner son axe : tu peux contrôler la vitesse du moteur et la direction de la rotation, mais guère plus. Pour maîtriser davantage le mouvement, tu as besoin de quelque chose qui te donne plus de contrôle : soit un servomoteur, soit un moteur pas-à-pas.

Il existe différents types de servomoteurs, mais le plus courant est celui qu'utilisent les amateurs de modélisme. Avec un moteur à courant continu, l'axe tourne, mais tu ne sais pas précisément sa position quand le moteur s'arrête. Un servomoteur sait en quelle position son axe est orienté. Bien que tu puisses lui indiquer dans quelle position tu souhaites qu'il s'arrête, il possède des limites : il ne peut s'arrêter que dans certaines positions et ne peut pas faire une rotation complète. Alors qu'un moteur à courant continu peut tourner de façon permanente, un servomoteur ne peut, en général, tourner que dans un angle de 180 degrés.

Un moteur pas-à-pas combine la puissance d'un moteur à courant continu et celle d'un servomoteur, en ce sens qu'il peut tourner sans s'arrêter et que tu peux lui indiquer de façon précise la position à laquelle tu souhaites qu'il s'arrête. Mais cela a un prix ! Les moteurs pas-à-pas ont tendance à être plus coûteux que les autres types de moteurs. Il existe une solution à ce problème : tu peux choisir le moteur à courant continu ou le servomoteur le moins cher pour ton projet (voir la figure 3-2) et contourner ses limites en créant toi-même ta propre solution.

Copyright © 2015 Eyrolles.

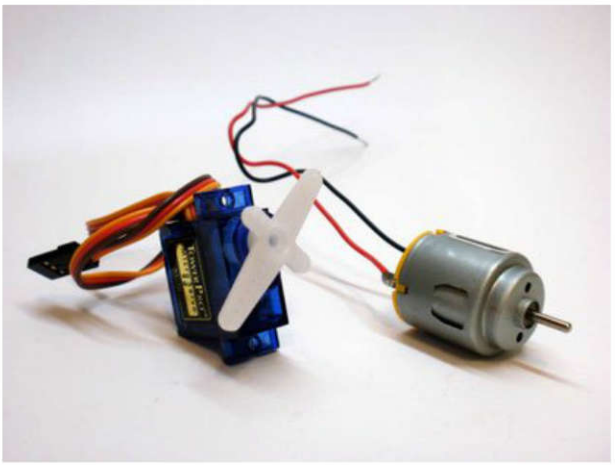


FIGURE 3-2 Un servomoteur et un moteur de jouet.

Un **servomoteur** est un moteur qui peut être contrôlé pour tourner jusqu'à une position spécifique. Normalement, il ne peut pas tourner de plus de 180 degrés.

Si tu souhaites en apprendre davantage sur l'utilisation des moteurs, tu peux lire le livre (en anglais) *Making Things Move* de Dustyn Roberts (Éditions McGraw-Hill, 2010).

Contrôle d'un servomoteur avec Arduino

L'environnement IDE d'Arduino comprend tout ce dont tu as besoin pour contrôler le fonctionnement d'un servomoteur. Il contient même des exemples de sketch pour t'aider à y parvenir. Afin de contrôler ton servomoteur, tu dois ouvrir le sketch nommé Sweep : lance l'IDE Arduino puis clique sur Fichier > Exemples > Servo > Sweeep (voir figure 3-3).

Ce sketch montre comment faire tourner un servomoteur. Regarde le code qu'il contient. La première ligne après les commentaires n'était pas dans les programmes que tu as utilisés dans les aventures 1 et 2.

```
#include <Servo.h>
```

Copyright © 2015 Eyrolles.

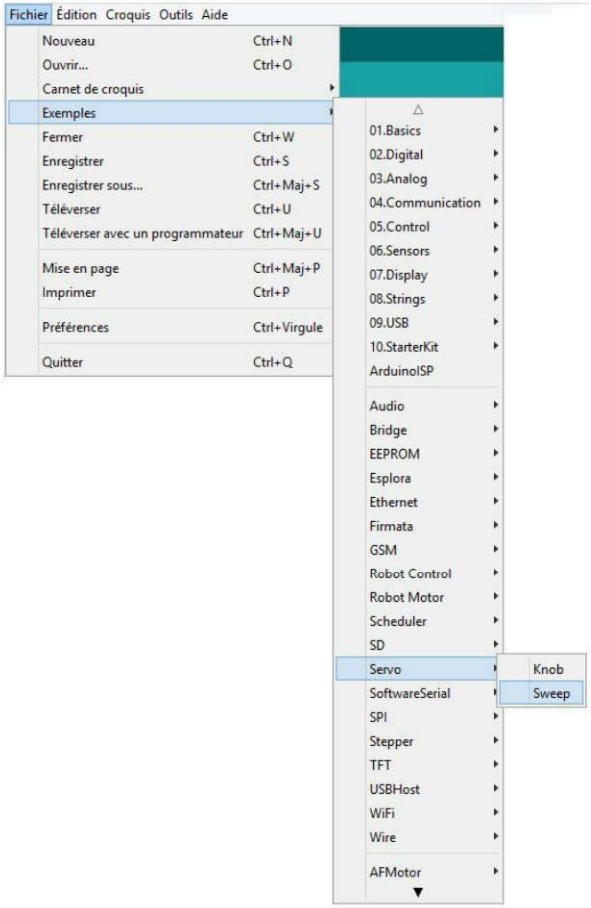


FIGURE 3-3 Ouverture du sketch Sweep.

L'instruction `#include` indique au compilateur Arduino que le sketch Sweep va utiliser certaines fonctions qui ne sont pas toujours inclues dans un sketch, et que le compilateur va les trouver dans une bibliothèque de fonctions appelée `Servo`. Ce code indique au compilateur qu'il doit d'abord lire le fichier de la bibliothèque `Servo.h`. Les caractères `<` et `>` encadrant le nom du fichier signifient que ce fichier est situé dans les dossiers standards de l'ordinateur où toutes les bibliothèques sont enregistrées.

Regarde maintenant la ligne de code suivante :

```
Servo myservo ;
```

Elle crée une nouvelle variable appelée `myservo`, mais cette variable n'est pas un entier comme celles que tu as déjà utilisées (par exemple, la variable `LEDverte` dans l'aventure 2). À la place, le type `Servo` (au lieu de `int`) est utilisé. Ce type `Servo` contient toutes les informations nécessaires pour communiquer avec un servomoteur.

Il ne reste plus qu'une ligne de code pour terminer de préparer le servomoteur dans le sketch Sweep. L'Arduino Uno à besoin de connaître la broche à laquelle le servomoteur sera

Copyright © 2015 Eyrolles.

connecté. Cette tâche ne doit être effectuée qu'une seule fois, donc elle se trouve dans la fonction `setup()`. Quelques lignes plus bas dans le sketch, tu devrais voir la ligne suivante :

```
myservo.attach(9) ;
```

Tu dois maintenant réaliser un circuit pour relier ton servomoteur à ta carte Arduino Uno. Il devrait ressembler à celui de la figure 3-4. Tu pourras constater qu'il ne nécessite que trois liaisons : +5 V, la masse GND et la broche de contrôle. La mauvaise nouvelle est que tous les servomoteurs n'ont pas ces connexions dans le même ordre. Certains (comme ceux des figures 3-1 et 3-2) possèdent un marquage qui repère les connexions. Si le tien n'en possède pas, renseigne-toi auprès de ton revendeur pour obtenir des informations à ce sujet. Si tu ne trouves aucune aide, tu peux essayer de connecter ton servomoteur de toutes les façons possibles jusqu'à ce qu'il fonctionne.

Réalise le circuit pour ton servomoteur.

1. Utilise un fil de pontage pour relier la broche GND (qui peut être repérée par GND ou 0 V) de ton servomoteur à l'une des broches GND de ta carte Arduino Uno.
2. Utilise un autre fil de pontage pour relier la broche 5 V du servomoteur à la broche +5 V de ta carte.
3. Utilise un troisième fil de pontage pour relier la broche restante de ton servomoteur à la broche 9 de ta carte.

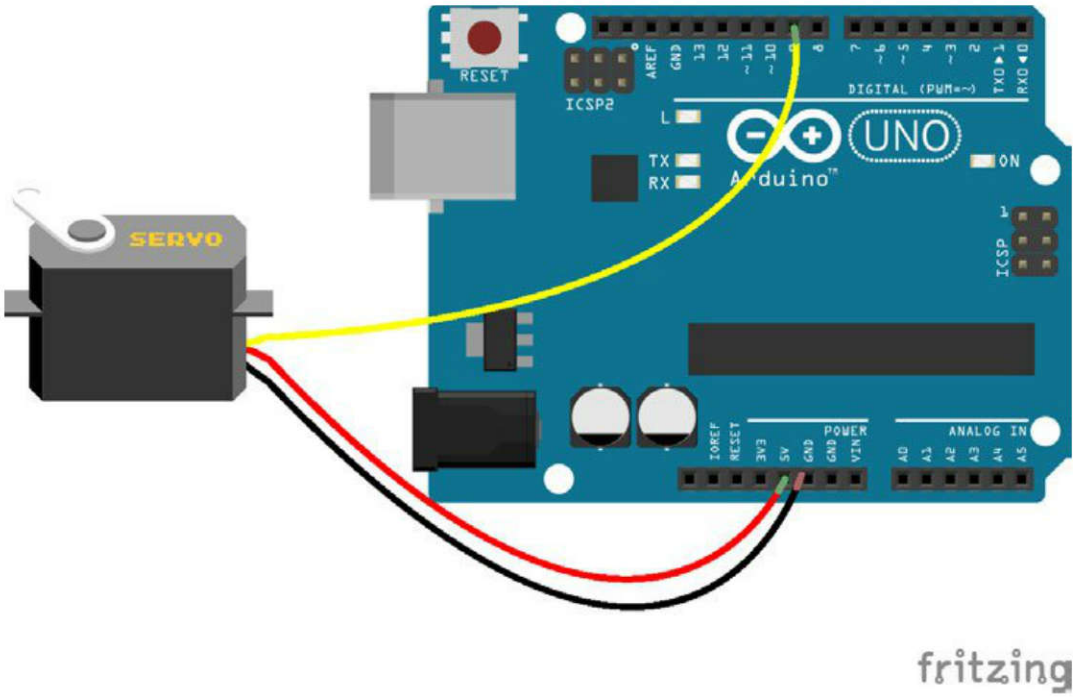


FIGURE 3-4 Le circuit reliant le servomoteur à ta carte Arduino Uno.

Copyright © 2015 Eyrolles.

Après avoir réalisé ce circuit, télécharge le sketch Sweep (retourne à l'aventure 1 si tu ne l'as pas déjà suivie) et regarde ton servomoteur bouger ! Il devrait tourner d'avant en arrière. Tu vas l'entendre et le voir travailler.

Répétition d'actions

La carte Arduino Uno contrôle le servomoteur par des impulsions électriques qui lui indiquent jusqu'où tourner son axe. Ne te préoccupe pas de savoir comment il le fait, la bibliothèque Servo s'en charge pour toi. Tout ce que tu dois faire, c'est indiquer au servomoteur où se positionner : tu peux choisir une position d'arrêt du servomoteur entre 0 et 180 degrés.

Dans le sketch Sweep, le servomoteur fait tourner son levier d'avant en arrière. Tu peux lui commander de faire cela en copiant les instructions `myservo.write` autant de fois que tu veux.

```
myservo.write(0) ;
myservo.write(1) ;
myservo.write(2) ;
myservo.write(3) ;
```

Ce n'est cependant pas la manière la plus efficace de procéder, car les ordinateurs sont très bons dans les tâches répétitives. Il existe en effet une meilleure façon de faire tourner le servomoteur d'avant en arrière. Si tu as déjà parcouru le code du sketch Sweep pour voir comment il est fait, tu as dû remarquer un élément de programmation appelé boucle `for`.

> Une boucle `for` est un moyen de programmer la répétition d'un bloc de code un nombre de fois prédéterminé.
>
>
>

Le sketch Sweep possède deux boucles `for`. Voici la première :

```
for(pos = 0; pos < 180; pos += 1)
{
  myservo.write(pos);
  delay(15);
}
```

Pour créer une boucle `for`, tu dois spécifier trois informations.

1. En premier, tu dois déterminer l'état du début de la boucle. Dans le sketch Sweep, c'est `pos = 0`.

Copyright © 2015 Eyrolles.

2. Ensuite, tu dois indiquer ce qui doit se produire pour que la boucle `for` continue. Dans notre cas, `pos` doit demeurer inférieure à `180` (`pos < 180` dans le code). Quand `pos` est égale ou supérieure à `180`, la boucle s'arrête et le code entre les crochets { } n'est plus exécuté.

3. Enfin, tu dois indiquer ce qui change à chaque fois que la boucle est exécutée. Dans le sketch Sweep, `1` est ajouté à `pos` chaque fois que le code entre { } est exécuté. Ceci est écrit `pos+=1`, de façon courte, mais tu peux l'écrire de différentes façons : `pos = pos + 1` ou `pos++`.

L'ordinateur ne tient pas compte des espaces ou retraits entre les instructions du code. Parfois, des espaces sont ajoutés dans le programme pour le rendre plus lisible, mais les deux lignes de code suivantes sont identiques pour Arduino.

```
for(pos=0; pos<180;pos+=1)
for ( pos = 0;  pos < 180 ; pos +=1 )
```

Il est plus facile de distinguer les trois parties d'une boucle `for` quand des espaces sont utilisés.

Ouf ! Il est temps de tout récapituler. Dans cet exemple, à l'intérieur de la boucle `for`, `pos` débute à `0`. Comme `0` est inférieur à `180`, le code entre les { } est exécuté. Le servomoteur est positionné à `0` et une pause de 15 millisecondes suit (en utilisant la fonction `delay()`). La valeur `1` est ensuite ajoutée à `pos`, qui vaut alors `1`. `1` étant plus petit que `180`, le servomoteur est positionné à `1` puis une pause de 15 millisecondes est à nouveau effectuée. Le programme continue ainsi jusqu'à ce que `pos` vaille `179`. Le servomoteur est alors positionné à `179` et `1` est ajouté à `pos`, lui donnant la valeur `180`. `pos` n'est alors plus inférieur à `180`, le code entre { } est ignoré et Arduino passe à la ligne de code qui suit immédiatement la boucle `for`.

DÉFI

Que se passe-t-il dans la seconde boucle `for` du sketch Sweep ? La voici :

```
for(pos = 180; pos>=1; pos-=1)
{
  myservo.write(pos);
  delay(15);
}
```

Modifie la boucle afin que le servomoteur ne tourne qu'entre les positions 0 et 90.

Copyright © 2015 Eyrolles.

Entrée numérique avec un bouton-poussoir

Tu penses peut-être qu'un **interrupteur** est un composant électrique simple. En réalité, les interrupteurs sont assez compliqués. Ils existent sous différentes formes et dimensions. Tu en as beaucoup chez toi pour allumer et éteindre les lumières : tout ce qu'ils font consiste à établir ou interrompre un courant électrique. Parfois, ils changent l'endroit où le courant électrique passe dans un circuit, mais le type d'interrupteur qui allume et éteint les lumières est constitué de deux parties métalliques qui se touchent ou ne se touchent pas, selon la position de l'interrupteur.

> Un **interrupteur** est un composant qui interrompt ou redirige le courant électrique dans un circuit.
>
>
>

Il existe un autre type d'interrupteur appelé **bouton-poussoir tactile**. Il est également constitué de deux parties métalliques, mais elles ne se touchent que lorsque le bouton est appuyé. Un bouton-poussoir tactile possède quatre broches, mais il est plus simple de considérer qu'il en a deux paires, car les deux broches d'une même paire sont reliées ensemble, même lorsque le bouton n'est pas appuyé. Quand on appuie sur le bouton, les quatre broches sont reliées. Regarde sur la figure 3-5 comment fonctionne un bouton-poussoir.

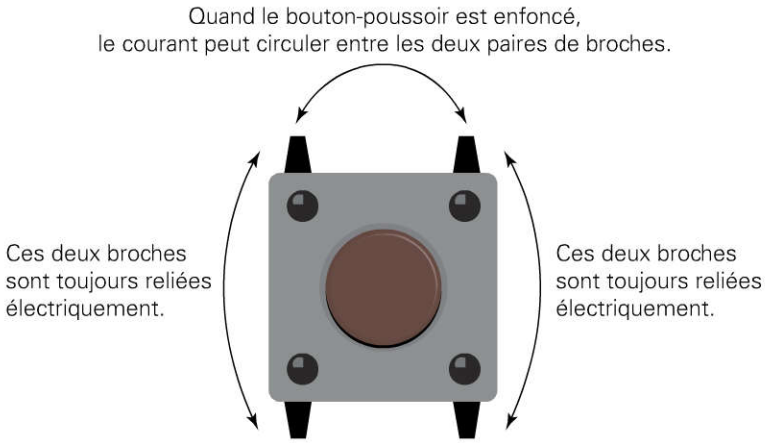


FIGURE 3-5 Le fonctionnement d'un bouton-poussoir.

Copyright © 2015 Eyrolles.

Un **bouton-poussoir tactile** est un type d'interrupteur particulier. Un bouton-poussoir à coupure interrompt le courant lorsqu'on appuie dessus. À l'inverse, un bouton-poussoir à contact ouvert n'interrompt le courant que s'il n'est pas enfoncé.

Maintenant, tu vas réaliser un circuit comprenant un bouton-poussoir, comme illustré sur la figure 3-6.

1. Insère le bouton-poussoir dans quatre rangées au centre de ta plaque de montage. Le bouton s'insère au-dessus de l'espace séparateur du milieu : ainsi, deux broches sont insérées dans la moitié supérieure de la plaque et les deux autres broches dans la partie inférieure.
2. Utilise un fil de pontage pour relier la rangée là où la broche en bas à droite du bouton-poussoir est insérée, dans l'une des longues rangées en bas de la plaque de montage. Si la rangée comporte une ligne bleue ou noire, ou un signe -, relie le fil à cette rangée ; si ta plaque ne comporte aucun marquage, relie le fil à n'importe laquelle des deux rangées en bas de la plaque.
3. Utilise un autre fil de pontage pour relier la rangée longue précédemment reliée au bouton-poussoir à l'une des broches GND de ta carte Arduino Uno.
4. Utilise un fil de pontage pour relier la broche 2 de ta carte à la rangée rattachée à la broche supérieure gauche de ton poussoir.

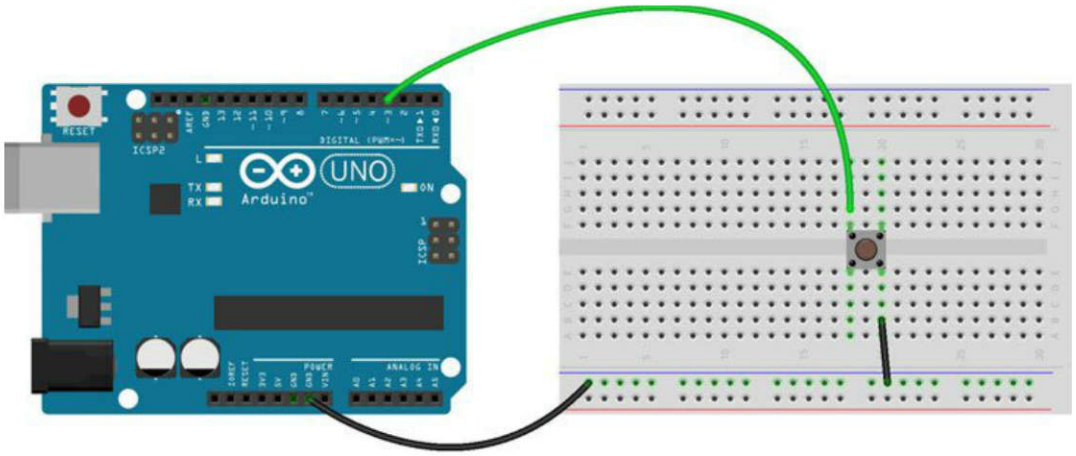


FIGURE 3-6 Le circuit avec bouton-poussoir tactile.

Maintenant que tu as réalisé ton circuit, ouvre le sketch situé dans les exemples : Fichier > Exemples > 01.Basics > DigitalReadSerial. Télécharge-le ensuite et ouvre le

Copyright © 2015 Eyrolles.

moniteur série en cliquant sur le bouton correspondant de l'IDE Arduino ou en cliquant sur le menu Outils > Moniteur série.

Tu es prêt ? Il est temps que tu appuies puis relâches le bouton-poussoir. Que se passe-t-il dans le moniteur série ? Quand ton doigt appuie sur le bouton, tu dois voir s'afficher le chiffre 0 ; quand le bouton est relâché, tu dois voir un mélange de 0 et de 1. La séquence de 0 et de 1 se fait au hasard, tu y verras peut-être une majorité de 0 ou une majorité de 1 plutôt qu'un mélange des deux. C'est ce que l'on appelle une **entrée flottante**. Quand le bouton n'est pas appuyé, la broche d'entrée d'Arduino n'est pas reliée à une source de tension de +5 V ni à la masse GND : elle est flottante. L'Arduino Uno reçoit alors des valeurs aléatoires de cette broche.

Une **entrée flottante** est une broche qui n'est reliée à rien. Cette broche lit des valeurs aléatoires si elle n'est pas reliée à un capteur ou à une source de tension telle que +5 V ou la masse GND.

Avoir des valeurs flottantes n'est pas un très bon principe. La raison principale est que quand Arduino reçoit un signal numérique sur l'une de ses broches, il « lit » la valeur 0 quand cette broche est reliée à la masse, 1 quand elle l'est au +5 V. Si la broche n'est reliée à rien, il lit aléatoirement des valeurs de 0 ou 1. Il est donc impossible pour ton programme de prendre les bonnes décisions basées sur la valeur de cette broche d'entrée. Si tu souhaites démarrer un moteur uniquement quand un bouton est pressé, reliant la broche à la masse GND, tu ne dois pas permettre à la broche de lire la valeur 0 si le bouton-poussoir n'est pas activé.

La façon de contourner ce problème consiste à utiliser une résistance de maintien (*pull-up resistor,* en anglais) au niveau +5 V, forçant ainsi un niveau haut par défaut sur la broche. Cette dernière est aussi reliée au bouton-poussoir, qui lui est lié à la masse GND. La résistance a en général une valeur élevée, par exemple 10 kΩ. Il n'y a aucune résistance entre la broche et la masse quand le bouton-poussoir est appuyé. La broche est alors reliée à la masse, au lieu de la tension +5 V, au travers de la résistance de maintien. L'électricité choisit toujours le chemin présentant le moins de résistance ; dans ce cas, c'est le chemin entre la broche et la masse GND.

Une **résistance de maintien** (*pull-up resistor*) est une résistance qui est reliée à la tension alimentant le circuit, établissant un niveau `HIGH` (HAUT) par défaut pour une broche. La résistance a habituellement une valeur de 10 kΩ.

Copyright © 2015 Eyrolles.

Comme souvent en électronique, l'effet d'une résistance de maintien est plus facile à comprendre quand tu construis un circuit et que tu regardes ce qui se passe. C'est ce que tu vas faire maintenant. Modifie ton circuit sur la plaque de montage pour le rendre identique à celui de la figure 3-7.

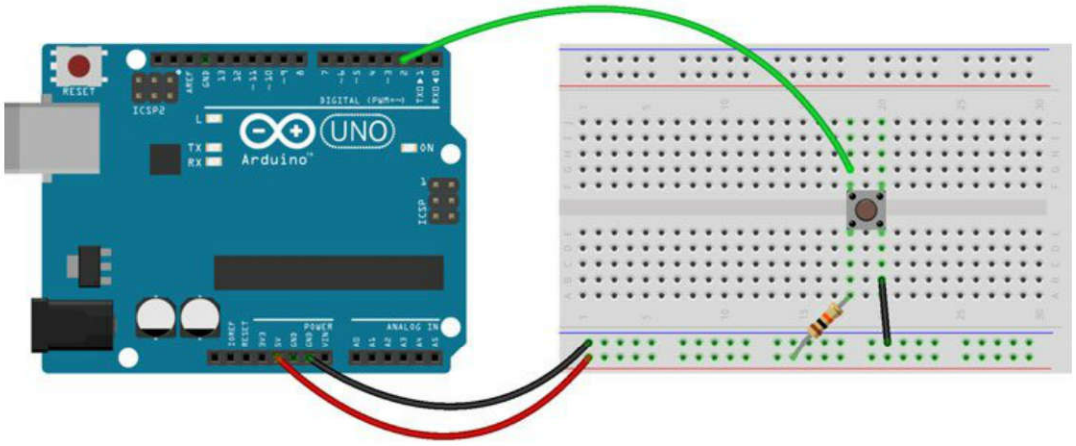


FIGURE 3-7 Le circuit avec une résistance de maintien.

1. Commence à partir du circuit que tu viens juste de réaliser avec le bouton-poussoir au centre de la plaque de montage.
2. Utilise un fil de pontage pour relier le +5 V de l'Arduino Uno à l'autre rangée longue en bas de la plaque (celle qui n'est pas liée à la masse GND).
3. Insère un fil de la résistance de 10 kΩ dans la rangée courte à laquelle est reliée la broche en bas et à gauche du bouton-poussoir. Insère l'autre fil de cette résistance dans la rangée longue, maintenant reliée au +5 V.

Tu n'as rien à modifier dans ton programme Arduino et tu peux laisser ouverte la fenêtre du moniteur série. Que se passe-t-il maintenant, quand tu appuies puis relâches le bouton-poussoir ? Tu ne devrais voir que des 0 quand le bouton est pressé et que des 1 lorsqu'il est relâché.

La carte Arduino possède déjà des résistances de maintien au niveau haut, tu peux donc les utiliser au lieu de les ajouter à ton circuit. Pour utiliser une résistance de maintien, tu dois l'indiquer lors de l'appel à la fonction `pinMode()` dans la fonction `setup()`, en écrivant les instructions suivantes :

```
pinMode(pushButton, INPUT_PULLUP) ;
```

Copyright © 2015 Eyrolles.

Modifie maintenant le sketch DigitalReadSerial afin que le second argument de la fonction `pinmode` soit `INPUT_PULLUP` au lieu de `INPUT`. La fonction `setup()` sera donc la suivante :

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the push button's pin an input:
  pinMode(pushButton, INPUT_PULLUP);
}
```

Pour terminer, télécharge le programme à nouveau et modifie le circuit sur ta plaque de montage selon celui de la figure 3-8 en retirant la résistance de 10 kΩ et le fil de pontage reliant le +5 V à l'une des rangées longues de la plaque de montage. Ton bouton-poussoir devrait agir de la même façon que lorsque tu y avais précédemment relié une résistance de maintien.

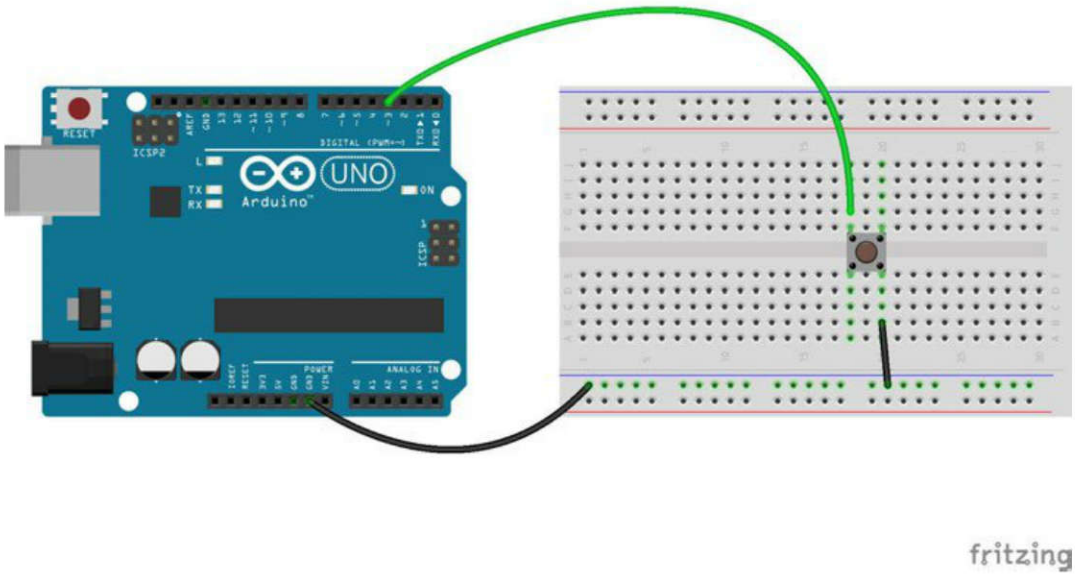


FIGURE 3-8 Le circuit avec un bouton-poussoir utilisant une résistance de maintien interne à la carte Arduino Uno.

Construction de ton coffre-fort à combinaison

Parfait ! Tu as déjà utilisé un bon nombre de capteurs et d'actionneurs. Tu peux maintenant les assembler pour réaliser un objet fantastique et très utile : un coffre à combinaison qui s'ouvre et se referme automatiquement (voir la figure 3-9). Pour l'ouvrir, tu utiliseras une combinaison et un bouton-poussoir. La boîte ne s'ouvrira que si la

Copyright © 2015 Eyrolles.

combinaison est correcte et elle restera ouverte tant que tu ne relâcheras pas le bouton-poussoir. Ce sera sûrement une cachette secrète pour tes bonbons ou pour éviter que tes crayons et stylos favoris ne soient « empruntés » sans ta permission.

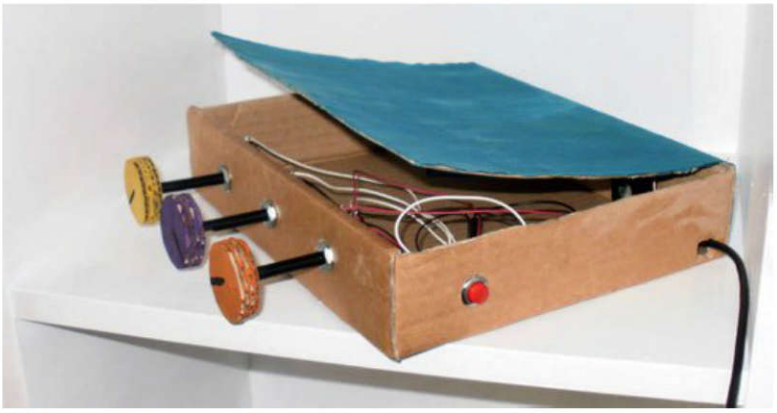


FIGURE 3-9 Le coffre à combinaison.

Ce dont tu as besoin

Tu as besoin des éléments suivants pour construire ton coffre. Cette liste comprend les composants nécessaires à la réalisation du prototype de ton circuit sur une plaque de montage et ceux dont tu auras besoin pour le coffre. La figure 3-10 montre les composants électroniques nécessaires.

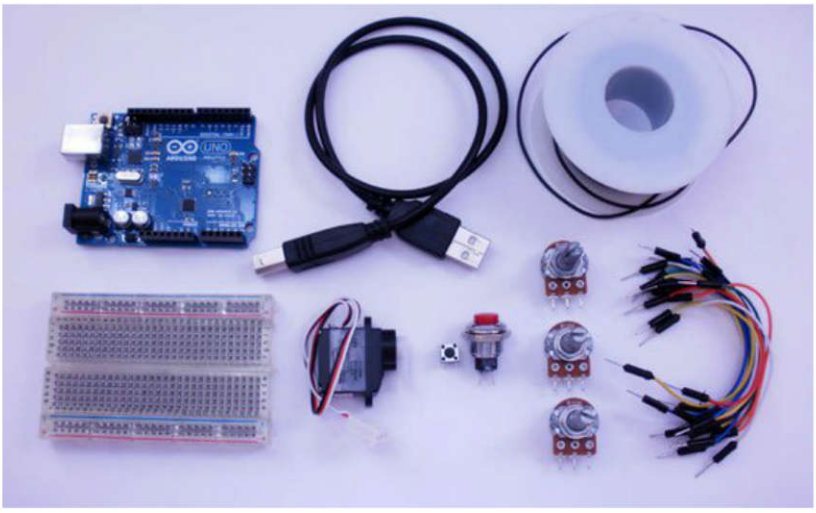


FIGURE 3-10 Les composants électroniques nécessaires à la réalisation du coffre-fort à combinaison.

Tu vas utiliser un type de bouton-poussoir différent pour ton coffre final et pour le prototype de ton circuit sur la plaque de montage. Ce sera en effet un **bouton-poussoir**

Copyright © 2015 Eyrolles.

pour montage en façade de boîtier, au lieu d'un bouton-poussoir tactile. Tu vas cependant toujours utiliser un bouton-poussoir tactile pour tester ton circuit sur la plaque de montage. Le bouton-poussoir prévu pour être monté en façade de boîtier est plus gros et plus facile à monter dans un coffret en carton. Tu verras comment relier les fils au coffret par la suite.

Un **bouton-poussoir pour montage en façade** est un bouton-poussoir qui est spécialement conçu pour être monté dans un boîtier. Il comporte un écrou et une rondelle prévus pour le maintenir en place sur la façade d'un coffre.

Procure-toi les fournitures suivantes avant de commencer ton projet :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 16 fils de pontage ;
- un bouton-poussoir tactile ;
- un bouton-poussoir pour montage en façade ;
- un servomoteur ;
- 3 potentiomètres de 10 kΩ ;
- de la soudure ;
- un fer à souder ;
- du fil de câblage ;
- un trombone ou un pic de brochette en bois ;
- une petite boîte avec un couvercle qui sera ton coffre-fort ;
- un pistolet à colle ;
- des ciseaux et un couteau.

Ta boîte peut avoir n'importe quelle taille, une boîte à chaussures conviendra parfaitement. Ce sera mieux si elle comporte un couvercle attaché à la boîte, mais si le couvercle est indépendant, je t'expliquerai comment l'attacher toi-même.

Tu peux utiliser ce que tu veux pour décorer ta boîte, par exemple de la peinture ou du papier.

Copyright © 2015 Eyrolles.

Va sur le site d'accompagnement pour voir une vidéo (en anglais) montrant comment construire le coffre-fort (www.editions-eyrolles.com/go/arduino).

Comprendre le circuit

Le circuit de ton coffre-fort comporte trois éléments principaux : un servomoteur, trois potentiomètres et un bouton-poussoir. Les trois potentiomètres sont reliés à trois entrées analogiques et le bouton-poussoir à une entrée numérique. Le servomoteur est contrôlé par une autre broche numérique.

La figure 3-11 montre le schéma du coffre-fort. Il te paraît compliqué, n'est-ce pas ? Ne t'en fais pas : tu vas le construire pas à pas. Avant de souder le circuit de ton coffre, tu vas en réaliser un prototype sur ta plaque de montage.

Dans les schémas électriques, les lignes indiquant les interconnexions se croisent souvent. Afin de rendre cela le plus clair possible, deux lignes qui sont électriquement reliées possèdent un petit cercle à leur intersection. Dans le cas contraire, il s'agit de deux fils qui ne sont pas électriquement en contact.

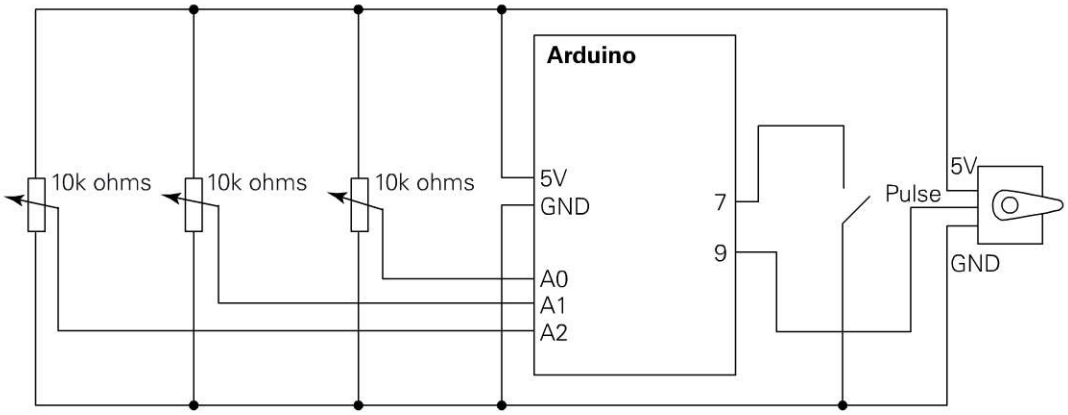


FIGURE 3-11 Le schéma du circuit du coffre-fort.

Le prototype sur plaque de montage

Tu devrais toujours réaliser le prototype d'un circuit avant d'en construire la version définitive. C'est quand même plus facile de corriger des erreurs avant d'avoir coupé les fils et soudé les liaisons ! Construis le circuit sur ta plaque de montage comme illustré sur la figure 3-12.

Copyright © 2015 Eyrolles.

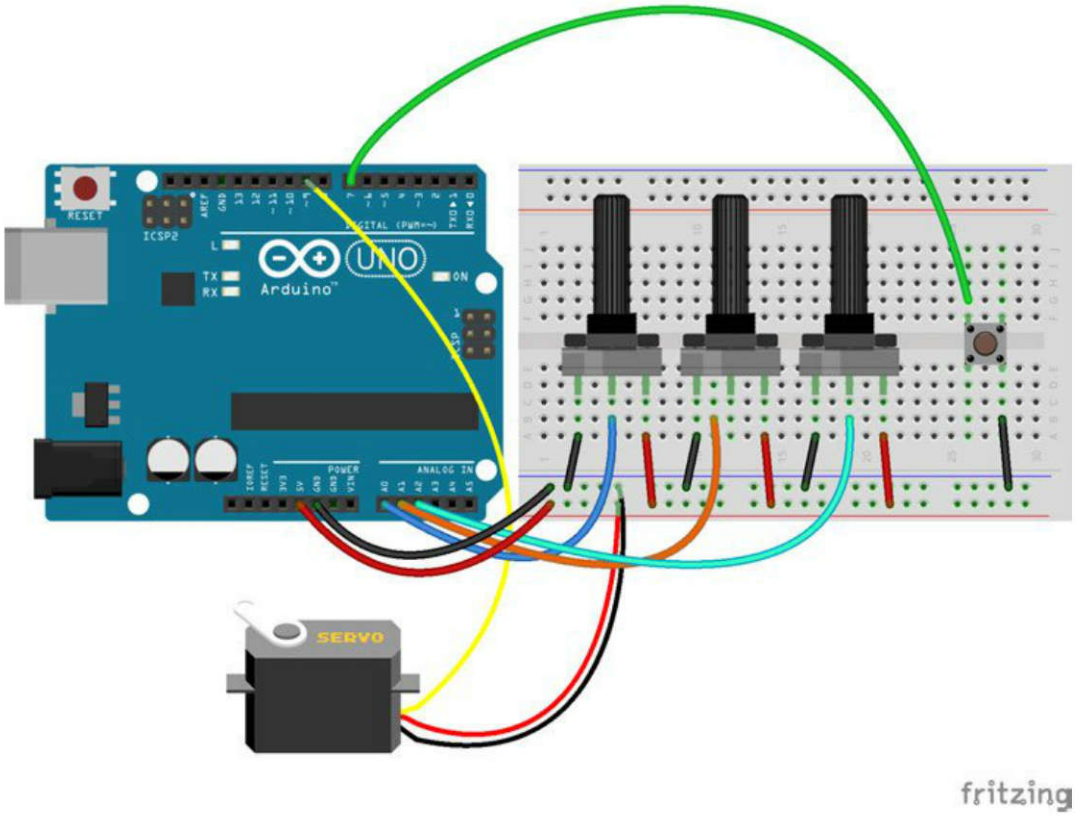


FIGURE 3-12 Le circuit du coffre-fort.

1. Commence par placer le bouton-poussoir tactile à droite de la plaque de montage. Il doit se trouver à cheval sur la séparation centrale de la plaque et chaque broche doit être insérée dans la rangée la plus proche.
2. Place les trois potentiomètres espacés régulièrement sur le reste de la plaque. Chaque broche des potentiomètres doit être insérée dans sa propre rangée de trous.
3. Utilise un fil de pontage pour relier la broche +5 V de ta carte Arduino Uno à l'une des rangées longues en bas de la plaque et un second fil pour relier sa broche GND à l'autre rangée longue. Si ta plaque est marquée (elles ne le sont pas toutes), relie le +5 V à la rangée identifiée par un trait rouge ou le signe +, et la masse GND à l'autre rangée munie d'un trait bleu ou noir ou du signe -.
4. Relie la broche gauche de chaque potentiomètre à la rangée longue connectée à la masse GND en utilisant trois fils de pontage.
5. Relie la broche droite de chaque potentiomètre à la rangée longue connectée à la broche +5 V, en utilisant trois autres fils de pontage.
6. Connecte la broche en bas et à droite du bouton-poussoir à la masse GND.
7. Utilise trois fils de pontage pour relier les broches du milieu de chaque potentiomètre aux broches A0, A1 et A2 de ta carte Arduino Uno.

Copyright © 2015 Eyrolles.

8. Utilise un fil de pontage pour relier la broche en haut et à gauche du bouton-poussoir à la broche 7 de l'Arduino Uno.

9. Connecte les trois fils du servomoteur. Relie la broche de masse du servomoteur à la rangée longue déjà reliée à la masse de la carte Arduino Uno, la broche +5 V à l'autre rangée longue et le fil restant à la broche 9 de la carte.

Écriture du programme

Tout comme le circuit, le programme du coffre-fort paraît compliqué à première vue. Mais en l'écrivant pas à pas, tu vas voir que ce sketch n'est que la combinaison de plusieurs petites parties de code. Le programme détecte si le bouton-poussoir est appuyé ou non. S'il l'est, la Uno vérifie si les trois potentiomètres sont positionnés correctement pour ouvrir le coffre. S'ils le sont, la boîte s'ouvre ; s'ils ne le sont pas, rien ne se passe.

Commence par lancer l'IDE Arduino. Ouvre un nouveau sketch avec les fonctions `setup()` et `loop()` vides :

```
void setup() {
}
void loop() {
}
```

En haut de ton sketch, avant l'appel à la fonction `setup()`, ajoute les lignes suivantes :

La version d'origine en anglais :

```
#include <Servo.h>

// The pins
int potPin1 = A0;
int potPin2 = A1;
int potPin3 = A2;
int buttonPin = 7;
int servoPin = 9;

// Other variables
int open1 = 0;
int open2 = 1023;
int open3 = 0;
int range = 10;
int boxOpen = 0;
Servo servo;
```

La version francisée :

```
#include <Servo.h>
```

Copyright © 2015 Eyrolles.

```
// Les broches
int brochePot1= A0;
int brochePot2= A1;
int brochePot3= A2;
int brocheBouton = 7;
int brocheServo = 9;

// Autres variables
int ouverture1 = 0;
int ouverture2 = 1023;
int ouverture3 = 0;
int ecart = 10;
// attention à ne jamais utiliser de lettres
// accentuées dans les instructions
int boiteOuverte = 0;
Servo servo;
```

La première ligne importe la bibliothèque permettant le contrôle du servomoteur, les lignes suivantes sont des déclarations de variables. Le premier groupe (après `// Pins` ou `// Broches`) contient les variables permettant de mémoriser les broches utilisées par les capteurs et l'actionneur.

Les cinq variables suivantes sont utilisées pour le contrôle de la boîte. Les variables `open1`, `open2` et `open3` ou `ouverture1`, `ouverture2` et `ouverture3` définissent les valeurs que les potentiomètres doivent retourner pour permettre l'ouverture de la boîte. Du fait de la difficulté de régler les potentiomètres à une valeur précise (surtout si tu n'utilises pas le moniteur série), la variable `range` ou `ecart` (sans accent), détermine de combien tu dois être précis dans ton réglage. Par exemple, toute valeur comprise entre `open2-range` ou `ouverture2-ecart` et `open2+range` ou `ouverture2+ecart` sera considérée comme correcte pour le second potentiomètre. Plus la valeur `range` ou `ecart` sera grande, plus il sera facile d'ouvrir la boîte.

La variable `boxOpen` ou `boiteOuverte` est utilisée pour mémoriser si la boîte est ouverte ou fermée. La boîte est d'abord fermée, la variable est donc positionnée à `0` au départ. Quand elle est ouverte, cette variable est positionnée à la valeur `1` et changée à nouveau à la valeur `0` quand elle est refermée. La fonction `loop()` contient le code qui contrôle la valeur de `boxOpen` ou `boiteOuverte`.

La dernière variable t'est familière : `servo`. C'est la variable qui permet de communiquer avec le servomoteur.

L'étape suivante consiste à ajouter le code dans la fonction `setup()` :

Code d'origine en anglais :

```
// set button pin to be an input with
// with pull-up resistor
```

Copyright © 2015 Eyrolles.

```
pinMode(buttonPin, INPUT_PULLUP);
// attach servo to pin
servo.attach(servoPin); // attaches the servo on pin 9
// to the servo object
servo.write(90); // start with the box closed
Serial.begin(9600); //start serial communication
```

Code francisé :

```
// la broche du poussoir est une entrée avec résistance de
// maintien
pinMode(brocheBouton, INPUT_PULLUP);
// attacher le servomoteur à sa broche
servo.attach(brocheServo); // attacher le servo à la broche 9
servo.write(90); // démarrer avec la boîte fermée
Serial.begin(9600); // démarrer la transmission série
```

La première ligne de la fonction `setup()` définit le mode du bouton-poussoir avec utilisation de la résistance de maintien au niveau haut. Le reste de la fonction attache le servomoteur à sa broche de contrôle, assure que la boîte est fermée par le servomoteur et démarre la transmission série.

Termine ton sketch en ajoutant les lignes suivantes dans la fonction `loop()` :

Code d'origine en anglais :

```
// check if button is pressed
int buttonValue = digitalRead(buttonPin);

// if button is pressed and box is closed
if(buttonValue == 0 && boxOpen == 0)
{
      // button is pressed
    int potValue1 = analogRead(potPin1);
    int potValue2 = analogRead(potPin2);
    int potValue3 = analogRead(potPin3);
    Serial.print("pot 1: ");

    Serial.print(potValue1);
    Serial.print(" pot 2: ");
    Serial.print(potValue2);
    Serial.print(" pot 3: ");
    Serial.println(potValue3);

      // if all values are within correct range
    if(potValue1 < (open1+range) &&
      potValue1 > (open1-range) &&
      potValue2 < (open2+range) &&
```

Copyright © 2015 Eyrolles.

```
      potValue2 > (open2-range) &&
      potValue3 < (open3+range) &&
      potValue3 > (open3-range))
        {
                        // open the box
          Serial.println("opening");
          for(int pos = 90; pos > 0; pos -= 1)
            {
            servo.write(pos);
            delay(15);
            }
          boxOpen = 1;
      }
}
// if button is released and box is open
if(buttonValue==1 && boxOpen==1)
{
    Serial.println("closing ");
     // close the box
     for(int pos = 0; pos < 90; pos+=1)
      {
        servo.write(pos);
        delay(15);
      }
    boxOpen = 0;
}
```

Code francisé :

```
// vérifions si le bouton est appuyé
int valeurBouton = digitalRead(brocheBouton);

// si le bouton est pressé et que la boîte est fermée
if(brocheBouton == 0 && boiteOuverte == 0)
{
  // le bouton est appuyé
  int valeurPot1 = analogRead(brochePot1);
  int valeurPot2 = analogRead(brochePot2);
  int valeurPot3 = analogRead(brochePot3);

  Serial.print("pot 1: ");
  Serial.print(valeurPot1);
  Serial.print(" pot 2: ");
  Serial.print(valeurPot2);
  Serial.print(" pot 3: ");
  Serial.println(valeurPot3);
```

Copyright © 2015 Eyrolles.

```
  // si toutes les valeurs sont dans les écarts permis
  if(valeurPot1< (ouverture1+ecart) &&
    valeurPot1> (ouverture1-ecart) &&
    valeurPot2< (ouverture2+ecart) &&
    valeurPot2> (ouverture2-ecart) &&
    valeurPot3< (ouverture3+ecart) &&
    valeurPot3> (ouverture3-ecart)
    )
      {

                        // ouvrir la boîte
           Serial.println("ouverture");
           for(int pos = 90; pos > 0; pos -= 1)
        {
             servo.write(pos);
             delay(15);
        }
        boiteOuverte = 1;
    }
}

// si le bouton est relâché et que la boîte est ouverte
if(valeurBouton==1 && boiteOuverte==1)
{
    Serial.println("fermeture ");
     // fermer la boîte
    for(int pos = 0; pos < 90; pos+=1)
    {
      servo.write(pos);
      delay(15);
    }
    boiteOuverte = 0;
}
```

Tu as maintenant terminé l'écriture de ton sketch et tu es prêt à vérifier que ton circuit fonctionne correctement. Télécharge ton sketch dans ta carte Arduino Uno et ouvre la fenêtre du moniteur série. Règle les trois potentiomètres pour qu'ils renvoient les valeurs mémorisées dans `open1`, `open2` et `open3` ou `ouverture1`, `ouverture2` et `ouverture3`. Appuie ensuite sur le bouton-poussoir et maintiens-le pressé. Le servomoteur devrait tourner puis s'arrêter. Relâche le bouton-poussoir, le servomoteur devrait revenir à sa position de départ.

L'encadré « Exploration du code » page 112 analyse plus en détail le code de la boucle pour expliquer comment fonctionne ton coffre-fort. Voici le sketch complet.

Copyright © 2015 Eyrolles.

La version originale en anglais :

```
#include <Servo.h>

// The pins
int potPin1 = A0;
int potPin2 = A1;
int potPin3 = A2;
int buttonPin = 7;
int servoPin = 9;

// Other variables
int open1 = 0;
int open2 = 1023;
int open3 = 0;
int range = 10;
int boxOpen = 0;

Servo servo;
void setup()
{
  // set button pin to be an input with
  // with pull-up resistor
  pinMode(buttonPin, INPUT_PULLUP);
  // attach servo to pin
  servo.attach(servoPin); // attaches the servo on pin 9
  //to the servo object
  servo.write(90); // start with the box closed
  Serial.begin(9600); //start serial communication
}

void loop()
{
  // check if button is pressed
  int buttonValue = digitalRead(buttonPin);
  // if button is pressed and box is closed
  if(buttonValue == 0 && boxOpen == 0)
  {
    // button is pressed
    int potValue1 = analogRead(potPin1);
    int potValue2 = analogRead(potPin2);
    int potValue3 = analogRead(potPin3);
    Serial.print("pot 1: ");
    Serial.print(potValue1);
    Serial.print(" pot 2: ");
    Serial.print(potValue2);
    Serial.print(" pot 3: ");
```

Copyright © 2015 Eyrolles.

```
    Serial.println(potValue3);

    // if all values are within correct range
    if(potValue1 < (open1+range) &&
      potValue1 > (open1-range) &&
      potValue2 < (open2+range) &&
      potValue2 > (open2-range) &&
      potValue3 < (open3+range) &&
      potValue3 > (open3-range)
      )
    {
      // open the box
      Serial.println("opening");
      for(int pos = 90; pos > 0; pos -= 1)
      {
            servo.write(pos);
            delay(15);
      }
      boxOpen = 1;
    }
  }

  // if button is released and box is open
  if(buttonValue==1 && boxOpen==1)
  {
    Serial.println("closing ");
    // close the box
    for(int pos = 0; pos < 90; pos+=1)
      {
           servo.write(pos);
           delay(15);
      }
    boxOpen = 0;
  }
}
```

La version francisée :

```
#include <Servo.h>

// Les broches
int brochePot1 = A0;
int brochePot2 = A1;
int brochePot3 = A2;
int brocheBouton = 7;
int brocheServo = 9;
```

Copyright © 2015 Eyrolles.

```
// Autres variables
int ouverture1 = 0;
int ouverture2 = 1023;
int ouverture3 = 0;
int ecart = 10;
int boiteOuverte = 0;
Servo servo;

void setup() {
// mettre ici le code d'initialisation qui sera exécuté
// une seule fois
// la broche du poussoir est une entrée avec résistance de
// maintien
pinMode(brocheBouton, INPUT_PULLUP);
// attacher le servo à sa broche
servo.attach(brocheServo); // attacher le servo à la broche 9
servo.write(90); // démarrer avec la boîte fermée
Serial.begin(9600); // démarrer la transmission série

}

void loop() {
  // vérifier si le bouton est appuyé
  int valeurBouton = digitalRead(brocheBouton);
  // si le bouton est pressé et que la boîte est fermée
  if(brocheBouton == 0 && boiteOuverte == 0)
  {
    // le bouton est appuyé
    int valeurPot1 = analogRead(brochePot1);
    int valeurPot2 = analogRead(brochePot2);
    int valeurPot3 = analogRead(brochePot3);

    Serial.print("pot 1: ");
    Serial.print(valeurPot1);
    Serial.print(" pot 2: ");
    Serial.print(valeurPot2);
    Serial.print(" pot 3: ");
    Serial.println(valeurPot3);

    // si toutes les valeurs sont dans les écarts permis
    if(valeurPot1< (ouverture1+ecart) &&
      valeurPot1> (ouverture1-ecart) &&
      valeurPot2< (ouverture2+ ecart) &&
      valeurPot2> (ouverture2- ecart) &&
      valeurPot3< (ouverture3+ ecart) &&
      valeurPot3> (ouverture3- ecart)
    )
    {
```

Copyright © 2015 Eyrolles.

```
      // ouvrir la boîte
      Serial.println("ouverture");
      for(int pos = 90; pos > 0; pos -= 1)
      {
           servo.write(pos);
           delay(15);
      }
      boiteOuverte = 1;
    }
  }

  // si le bouton est relâché et que la boîte est ouverte
  if(valeurBouton==1 && boiteOuverte==1)
  {
    Serial.println("fermeture ");
    // fermer la boîte
    for(int pos = 0; pos < 90; pos+=1)
      {
           servo.write(pos);
                 delay(15);
      }
    boiteOuverte = 0;
  }
}
```

EXPLORATION DU CODE

Regardons un peu plus en détail le code que tu viens d'entrer dans la fonction `loop()`.

La valeur de `brocheBouton` est lue. Si la valeur est 0 et que la boîte est fermée, alors les valeurs de chaque potentiomètre sont lues :

```
// vérifier si le bouton est appuyé
  int valeurBouton = digitalRead (brocheBouton);
// si le bouton est pressé et que la boîte est fermée
  if(brocheBouton == 0 && boiteOuverte ==0) {
// le bouton est appuyé
  int valeurPot1 = analogRead(brochePot1);
  int valeurPot2 = analogRead(brochePot2);
  int valeurPot3 = analogRead(brochePot3);
```

La valeur de chaque potentiomètre est alors affichée dans le moniteur série pour aider à la mise au point du programme.

Copyright © 2015 Eyrolles.

```
  Serial.print("pot 1: ");
  Serial.print(valeurPot1);
  Serial.print(" pot 2: ");
  Serial.print(valeurPot2);
  Serial.print(" pot 3: ");
  Serial.println(valeurPot3);
```

Si chaque potentiomètre renvoie une valeur dans les limites acceptables :

```
// si toutes les valeurs sont dans les écarts permis
  if(valeurPot1< (ouverture1+ecart) &&
    valeurPot1> (ouverture1-ecart) &&
    valeurPot2< (ouverture2+ecart) &&
    valeurPot2> (ouverture2-ecart) &&
    valeurPot3< (ouverture3+ecart) &&
    valeurPot3> (ouverture3-ecart)
    ) {
```

Alors la boîte est ouverte en utilisant une boucle `for` pour faire tourner le servomoteur à la position `0`. Tu sais que la boîte est fermée si `boiteOuverte` vaut `0` `0`. Après que la boîte soit ouverte, `boiteOuverte` est mise à la valeur `1`, ce qui confirme que la boîte est ouverte.

```
    // ouvrir la boîte
    Serial.println("ouverture");
    for(int pos = 90; pos > 0; pos -= 1)
      {
        servo.write(pos);
        delay(15);
      }
    boiteOuverte = 1;
    }
  }
```

Si la valeur de `brocheBouton` est de `1` et que la boîte est ouverte, celle-ci est refermée en utilisant une boucle `for` pour faire tourner le servomoteur à la position `90`. La variable `boiteOuverte` est alors changée à `0`.

```
// si le bouton est relâché et que la boîte est ouverte
  if(valeurBouton==1 && boiteOuverte==1) {
    Serial.println("fermeture ");
    // fermer la boîte
    for(int pos = 0; pos < 90; pos+=1)
      {
        servo.write(pos);
        delay(15);
      }
```

Copyright © 2015 Eyrolles.

```
        boiteOuverte = 0;
      }

    }
```

Si le bouton-poussoir est appuyé alors que la boîte est déjà ouverte ou relâché quand la boîte est déjà fermée, il ne se passe rien et la fonction `loop()` est répétée.

DÉFI

Définis ta combinaison secrète dans les variables `ouverture1`, `ouverture2` et `ouverture3`. Ajuste ensuite la variable `ecart` pour rendre facile le positionnement des potentiomètres à la bonne combinaison.

Réalisation du coffre-fort

Tu es enfin prêt à construire ton coffre-fort ! Le processus est assez similaire à la construction de la boîte d'affichage des messages de l'aventure 2. Tu vas utiliser une boîte (à chaussures par exemple) pour y loger le circuit électronique.

1. Attache le couvercle à la boîte. Assure-toi de l'attacher à un seul côté, afin que la charnière puisse ouvrir et fermer le couvercle. De cette façon, le servomoteur pourra ouvrir le couvercle en le poussant sans le faire tomber. Pour y parvenir, fabrique une charnière avec une bande de papier à coller, comme le montre la figure 3-13.

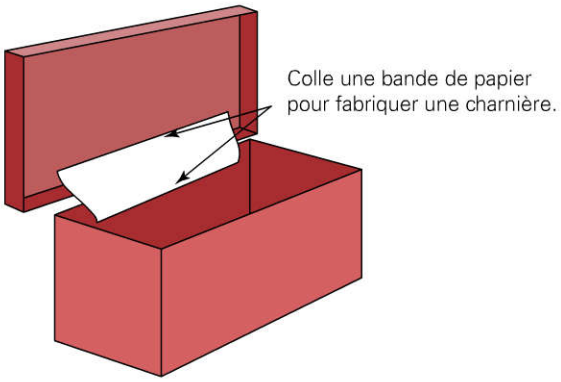


FIGURE 3-13 Si le couvercle n'est pas déjà attaché à ta boîte, ajoute une charnière en papier.

Copyright © 2015 Eyrolles.

2. Ensuite, tu dois décider de l'endroit où tu veux mettre tes potentiomètres et ton bouton-poussoir. Tu peux les mettre où tu veux, mais tu choisiras probablement de les placer sur le devant de la boîte, pour pouvoir y accéder facilement. Perce les trous nécessaires au passage de l'axe des potentiomètres et du bouton de manière à ce qu'ils s'insèrent parfaitement. Perce également un trou pour le passage du câble USB, qui alimentera ta carte Arduino Uno.

3. Les servomoteurs sont munis de différents leviers. Ils s'insèrent à l'extrémité de l'axe rotatif du servomoteur. Tu vas utiliser celui qui est constitué d'un seul bras partant de l'axe (n'utilise pas celui qui est muni de quatre bras). Le bras n'est pas très long, tu dois donc l'allonger en lui attachant un autre objet, comme un pic à brochettes en bois ou un grand trombone. Colle cette rallonge au bras du servomoteur et vérifie qu'elle est solidement attachée (voir la figure 3-14).

4. Télécharge le sketch du coffre-fort dans ta carte Arduino Uno et assemble le circuit sur ta plaque de montage, si tu ne l'as pas déjà fait.

5. Retire le levier de ton servomoteur. Si le programme fonctionne et que tu n'appuies pas sur le bouton-poussoir, le servomoteur devrait se positionner à 90 degrés. Attache maintenant le levier afin qu'il soit sur la position 90 degrés, pour qu'il ne pousse pas le couvercle en position ouverte. En plaçant les potentiomètres selon ta combinaison puis en appuyant sur le bouton-poussoir, le servomoteur devrait positionner son bras vers le haut.

6. La rallonge du bras du servomoteur va ainsi ouvrir le couvercle, mais comment le fermera-t-il après l'avoir ouvert ? Confectionne une boucle en papier collé à l'intérieur du couvercle comme sur la figure 3-15. L'extension du bras doit entrer dans la boucle du couvercle afin de le ramener en position fermée.

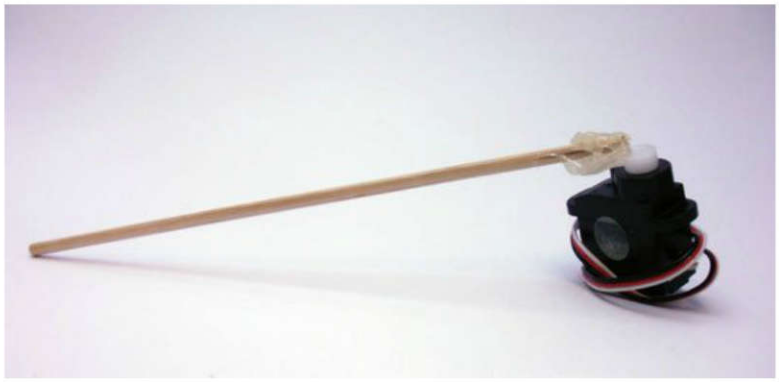


FIGURE 3-14 Allonge le bras du servomoteur avec un pic à brochettes ou un grand trombone.

Copyright © 2015 Eyrolles.

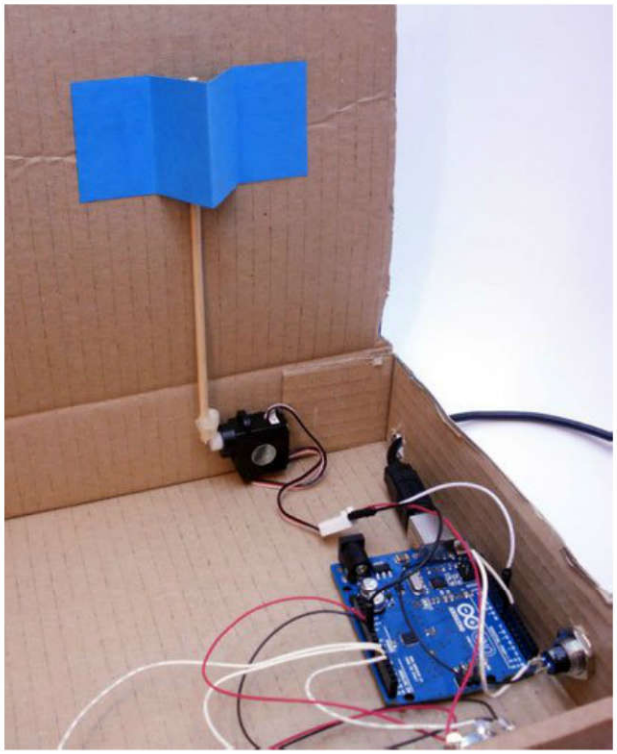


FIGURE 3-15 La boucle de papier permettant au servomoteur de refermer le coffre-fort.

Soudage des fils

Suis les étapes suivantes pour souder les fils.

1. Place les potentiomètres dans les trous de la boîte. Mesure et coupe quatre longueurs de fil afin de relier le potentiomètre le plus éloigné de la carte Arduino Uno au potentiomètre suivant. Coupe les fils environ 2,5 cm plus longs que la longueur nécessaire. Dénude à peu près 12 mm aux extrémités de chacun. Répète cette étape pour découper les fils qui serviront à relier le potentiomètre du centre à celui qui est le plus proche de la carte Arduino Uno (voir la figure 3-16). Ces fils vont servir à relier les broches externes de chaque potentiomètre entre elles.
2. Coupe deux longueurs de fils qui iront des broches externes du potentiomètre le plus proche de la Uno aux broches +5 V et GND de la carte. Coupe les fils environ 2,5 cm plus longs que la longueur nécessaire et dénude-les sur environ 12 mm à chaque extrémité.
3. Coupe un morceau de fil qui ira du potentiomètre le plus proche de la carte Arduino Uno au servomoteur. Coupe-le environ 2,5 cm plus long que la longueur nécessaire et dénude-le sur environ 12 mm à chaque extrémité.

Copyright © 2015 Eyrolles.

Souviens-toi que tu dois effectuer les soudures sous le contrôle d'un adulte. Va sur le site `www.sonelec-musique.com/electronique_bases_tutoriel_soudure.html` pour apprendre à souder.

4. Soude ensemble les broches externes des potentiomètres les plus éloignés de l'Arduino Uno en utilisant les fils comme sur la figure 3-16.

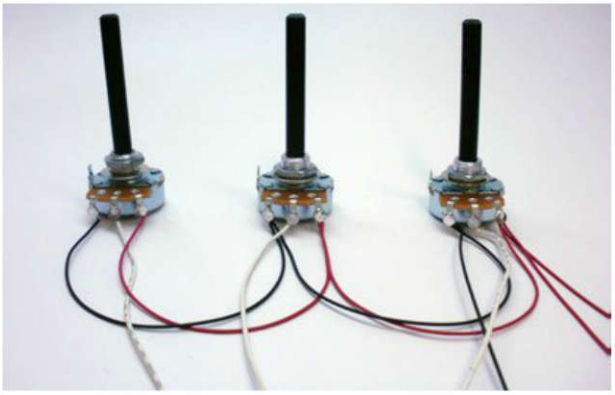


FIGURE 3-16 Soudure des composants.

5. Soude les fils du potentiomètre le plus proche de la carte Arduino Uno. L'une des broches externes comportera deux fils : l'un venant du potentiomètre du centre et l'autre allant vers la carte Arduino Uno. Trois fils seront soudés à l'autre broche externe de ce potentiomètre : l'un provenant du potentiomètre central, un autre provenant de la carte Arduino Uno et le dernier étant relié au servomoteur.
6. Mesure et coupe trois morceaux de fil qui relieront chacun des potentiomètres aux broches A0, A1 et A2 de la carte Arduino Uno. Coupe les fils environ 2,5 cm plus longs que la longueur nécessaire et dénude à peu près 12 mm aux extrémités de chaque fil.
7. Soude une extrémité de chaque fil à la broche centrale de chaque potentiomètre.
8. Place le bouton-poussoir dans son trou. Mesure et coupe deux morceaux de fil qui relieront les broches GND et sept broches de la carte Arduino Uno. Coupe les fils environ 2,5 cm plus longs que la longueur nécessaire et dénude à peu près 12 mm aux extrémités de chacun.
9. Soude un de ces fils à une des broches du bouton-poussoir et l'autre à la deuxième broche.

Voici un petit résumé, pour simplifier.

- Tu vas avoir trois fils provenant de la broche centrale de chaque potentiomètre qui seront reliés à la carte Arduino Uno.

Copyright © 2015 Eyrolles.

- Le potentiomètre le plus éloigné de la carte Arduino Uno aura un fil reliant chacune de ses broches externes à une des broches externes du potentiomètre du centre.
- Le potentiomètre central aura deux fils connectés à chacune de ses broches externes : l'un sera relié au premier potentiomètre et le second au dernier.
- Le dernier potentiomètre aura deux fils reliés à l'une de ses broches externes et trois fils reliés à l'autre. La broche avec deux fils sera reliée au potentiomètre central et à la carte Arduino Uno. La broche avec trois fils sera reliée au potentiomètre du milieu, au servomoteur et à la carte Arduino Uno.
- Le bouton-poussoir aura un fil soudé à chaque broche. Ces fils seront reliés à la carte Arduino Uno.

Chaque boîte est différente. Sa taille ne sera pas la même et le couvercle sera plus ou moins fermement attaché. Tu devras faire preuve de créativité pour résoudre tous les problèmes techniques de manière à ce que ta boîte s'ouvre et se referme correctement. Par exemple, tu pourras avoir besoin de coller une boucle en papier à l'intérieur du couvercle pour permettre au bras du servomoteur de l'ouvrir et le refermer (regarde la figure 3-15).
Pourquoi n'essaierais-tu pas de fabriquer ta propre boîte en papier ? Procure-toi un livre sur l'origami ou cherche sur Internet un plan pour réaliser une boîte à couvercle.

Insertion de l'électronique

Les potentiomètres et le bouton-poussoir sont équipés de rondelles et d'écrous qui se vissent sur la base de leurs axes. Retire les écrous et les rondelles et insère les tiges des potentiomètres et du bouton-poussoir dans les trous de ta boîte. Insère les rondelles et visse les écrous pour maintenir en place ces composants sur la boîte. Ajoute un peu de colle s'ils remuent plus que prévu, mais fais bien attention de ne pas laisser la colle couler sur les axes des composants !

Dans la boîte, réalise ton circuit en reliant les fils à leurs broches respectives de la carte Arduino Uno. Un des fils soudés à l'une des broches externes du potentiomètre (celle qui comporte trois fils) doit être inséré dans la broche +5 V ; un des fils venant de l'autre broche (celle qui comporte deux fils) sera relié à la broche GND. Connecte les trois fils provenant des potentiomètres aux broches A0, A1 et A2. Relie un des fils du bouton-poussoir à la broche GND et l'autre à la broche 7. Relie enfin le dernier fil du potentiomètre comportant trois fils à la liaison +5 V du servomoteur. Utilise deux fils de pontage pour relier le servomoteur aux broches GND et 9 de la carte Arduino Uno.

Tu dois maintenant fixer ton servomoteur à l'intérieur de la boîte, sur le flanc auquel est fixée la charnière du couvercle. Prends bien soin de vérifier qu'il permet d'ouvrir et refermer correctement la boîte, sans forcer ni laisser son bras s'échapper de la boucle

Copyright © 2015 Eyrolles.

en papier. Il sera peut-être nécessaire de modifier la taille et la position de la boucle en papier, si tu ne parviens pas à obtenir un fonctionnement correct du couvercle en ajustant simplement la position du servomoteur. Quand sa position sera déterminée, tu devras coller le servomoteur sur le côté intérieur de la boîte. Tu pourras confectionner une boucle étrier en papier, en forme de U, autour du servomoteur pour renforcer sa fixation au boîtier. Mais surtout, fais attention de ne pas laisser la colle aller sur l'axe de ton servomoteur.

Maintenant, teste le tout ! Tu dois avoir un coffret avec un code secret qui s'ouvre automatiquement. La figure 3-17 montre le câblage terminé.

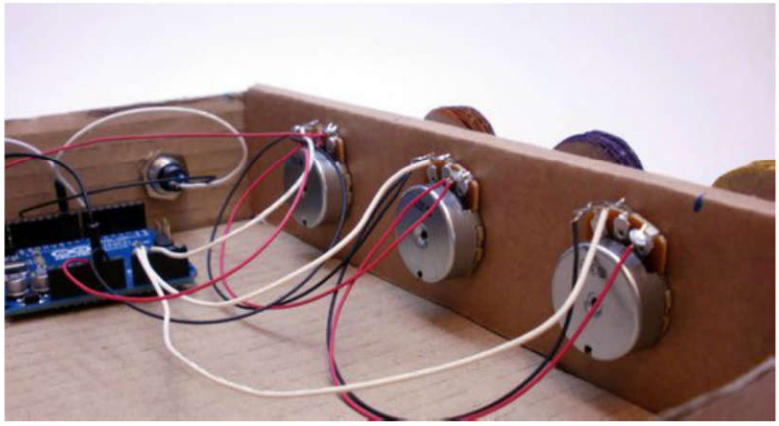


FIGURE 3-17 Le coffre-fort terminé.

Plus d'aventures avec Arduino

Maintenant que tu as l'expérience de l'utilisation des servomoteurs, regarde ces différents projets :

- `http://playground.arduino.cc/ComponentLib/Servo`
- `http://arduino.cc/en/pmwiki.php?n=Tutorial/Knob`

L'utilisation de servomoteurs avec un microcontrôleur Arduino n'est que le début de ce que tu peux faire dans le domaine de la robotique. Voici différents robots, dont la plupart ont été réalisés avec un Arduino !

- `http://artbots.org/2011/participants/`
- `http://makezine.com/projects/building-a-simplearduino-robot/`
- `http://www.makershed.com/collections/robotics`

Copyright © 2015 Eyrolles.

Rappel des commandes Arduino de cette aventure	
Commande	**Description**
`#include`	Commande permettant d'importer une bibliothèque de code. Voir également `http://arduino.cc/en/Reference/Include`.
`Servo.h`	Bibliothèque de fonctions permettant le contrôle d'un servomoteur. Voir également `http://arduino.cc/en/Reference/servo`.
`Servo`	Objet permettant le contrôle d'un servomoteur. Voir également `http://arduino.cc/en/Reference/servo`.
`Servo.attach()`	Permet d'affecter une broche de contrôle à la variable identifiant un servomoteur. Voir également `http://arduino.cc/en/Reference/ServoAttach`.
`Servo.write()`	Envoie une valeur à un servomoteur pour lui indiquer la position qu'il doit atteindre. Voir également `http://arduino.cc/en/Reference/ServoWrite`.
`for`	Boucle permettant à une partie d'un programme de se répéter un certain nombre de fois. Voir également `http://arduino.cc/en/Reference/For`.

Étape franchie : Tu sais désormais combiner circuits et code !

Dans la prochaine aventure

Quelques LED c'est bien, mais plein de LED c'est encore mieux ! Dans la prochaine aventure, tu apprendras à contrôler une grande quantité de LED à l'aide d'un petit circuit intégré appelé « registre à décalage ».

Copyright © 2015 Eyrolles.

Aventure 4
Utilise des registres à décalage

TU ES EN BONNE VOIE pour devenir un expert en Arduino. Tu as abordé toutes sortes de composants, des moteurs aux potentiomètres. Tu as même géré trois potentiomètres simultanément. Mais comment interagir avec plus de trois éléments de la même sorte ? Une LED c'est bien, trois c'est mieux, mais comment faire avec vingt-quatre ?

En tant que spécialiste Arduino, en regardant ta carte, tu pourrais te poser des questions sur ma capacité à compter. Il n'y a pas vingt-quatre broches de sortie pour contrôler des LED sur ta carte ? Tu as raison ! Mais tu peux t'en remettre à la puissance d'un circuit intégré spécial appelé registre à décalage pour augmenter le nombre de sorties. C'est ce que tu vas découvrir dans cette aventure.

Quand tu commences à travailler avec d'aussi nombreuses sorties, le code devient vite désordonné. Je vais donc te montrer comment le garder propre et facile à comprendre.

En assemblant le tout, à la fin de cette aventure, tu auras créé un affichage lumineux de ton nom (ou de tout autre mot de ton choix), à la manière d'une banderole de carnaval.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une grande plaque de montage ou 2 petites ;
- 38 fils de pontage ;
- 16 LED ;
- 16 résistances de 220 Ω ;
- 2 registres à décalage 74HC595 (circuits intégrés).

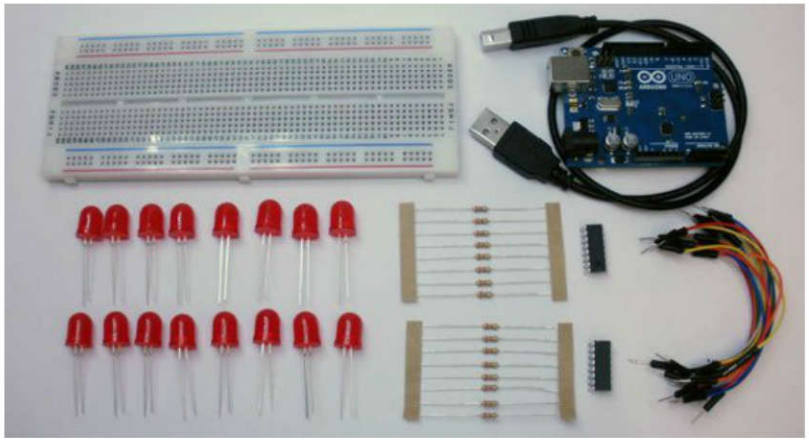


FIGURE 4-1 Les éléments nécessaires pour la première partie de cette aventure.

Les noms des circuits intégrés sont parfois longs et compliqués, mais en réalité ils ne contiennent que quelques éléments d'information. Pour cette aventure, tu vas utiliser un registre à décalage 74HC595, ces chiffres et lettres constituent la référence du circuit, c'est tout ce que tu dois retenir pour acheter ce composant. Si tu achètes un circuit intégré qui comporte deux lettres additionnelles avant 74HC595, c'est idéal. Ces lettres sont un code identifiant le fabricant du circuit intégré. Les registres à décalage sont fabriqués par de nombreuses sociétés, si ces lettres apparaissent avant la référence du circuit intégré, tu n'as donc pas à t'en soucier. Tant que le composant que tu achètes possède la suite de chiffres et lettres 74HC595, c'est parfait. Il peut aussi y avoir une lettre additionnelle à la fin de cette référence, elle indique le type de boîtier du circuit intégré. Pour les plaques de montage, tu dois avoir un boîtier N ou DIP (un composant possédant deux rangées de broches qui s'insèrent dans la plaque de montage). La section « Comment obtenir plus de sorties avec un registre à décalage ? », page 130, fournit davantage de précisions à ce sujet.

Copyright © 2015 Eyrolles.

Organisation de ton code

Un programme est une simple suite d'instructions qu'un ordinateur peut comprendre. Ainsi, tu peux gagner du temps en copiant la même partie de code plusieurs fois pour que l'ordinateur répète le même jeu d'instructions. Mais alors que se passe-t-il si tu commets une erreur typographique ? Si tu oublies, par exemple, un point-virgule ? La plus petite erreur peut amener ton programme à ne pas fonctionner correctement. Il peut être difficile de comprendre pourquoi il fonctionne les deux premières fois, mais pas la troisième. De longues parties de code répétées rendent encore plus difficile le suivi de ce qui se passe dans un sketch, car ton code devient alors illisible.

Les programmeurs aiment à dire qu'ils sont paresseux et qu'ils n'aiment pas faire plus de travail que nécessaire ! Ainsi, les experts en ordinateurs qui ont créé les langages de programmation ont passé beaucoup de temps à définir la façon dont les instructions sont écrites, de manière à minimiser le risque de commettre de simples erreurs en faisant des copier-coller de parties de code. La section suivante fournit quelques techniques que tu peux utiliser pour simplifier ton programme.

Utilisation des fonctions

Une façon pratique de dupliquer le code que tu souhaites répéter consiste à le placer dans ce que l'on appelle une fonction. Cela consiste à donner un nom à un jeu d'instructions. Tu n'as alors qu'à écrire le nom de ce groupe d'instructions chaque fois que tu souhaites l'exécuter, au lieu de le réécrire entièrement.

Si tu as lu les aventures précédentes, tu as déjà utilisé des fonctions écrites par quelqu'un d'autre. Par exemple, `digitalWrite()` est une fonction qui contrôle une broche de sortie numérique de ta carte Arduino Uno. La fonction prend en charge tous les détails nécessaires pour activer ou désactiver la broche. Quant à toi, tu n'as plus qu'à l'appeler. Te voici maintenant prêt à écrire tes propres fonctions.

Dans cette aventure, tu vas ajouter des fonctions au sketch `Blink`. Tu dois te rappeler que j'ai déjà parlé des fonctions, en particulier `setup()` et `loop()`. Elles sont identiques à celles que tu vas utiliser dans le sketch `Blink`, cependant tu ne devras pas leur donner toi-même un nom : elles devront s'appeler obligatoirement `setup()` et `loop()`. En effet, quand Arduino démarre, elle cherche d'abord la fonction `setup()` et exécute les lignes de code qu'elle contient. Elle cherche ensuite la fonction `loop()` et exécute de façon répétitive les lignes de code qu'elle contient.

Copyright © 2015 Eyrolles.

Il est toujours plus facile de comprendre un nouveau concept en le mettant en œuvre par soi-même. Aussi retourne voir le contenu du sketch Blink avec lequel tu as déjà travaillé dans l'aventure 1. Ouvre-le à partir du menu Fichier > Exemples > 01.Basics > Blink.

En premier lieu, tu vas enregistrer une copie de ce sketch, plutôt que d'utiliser l'original, car tu vas le modifier. Ceci évitera de changer le sketch original des exemples. Enregistre ta copie en passant par le menu Fichier > Enregistrer sous... et donne-lui le nom `BlinkFonctions.ino`.

Examine ton nouveau sketch BlinkFonctions.ino (ta copie de l'exemple Blink). La plupart de ses actions se situent dans la fonction `loop()`.

Le code original :

```
// The loop routine runs over and over again forever :
void loop() {
  digitalWrite(led, HIGH); // turn the LED on
  delay(1000); // wait for a second
  digitalWrite(led, LOW); // turn the LED off
  delay(1000); // wait for a second
}
```

Le code francisé :

```
// La fonction loop s'exécute indéfiniment :
void loop() {
  digitalWrite(led, HIGH); // allume la LED
  delay(1000); // pause d'une seconde
  digitalWrite(led, LOW); // éteint la LED
  delay(1000); // pause d'une seconde
}
```

Tu peux créer ta propre fonction effectuant la même chose que les quatre lignes de code de la fonction `loop()`. Mais avant de commencer, arrête-toi un instant ! Il y a deux choses importantes que je veux d'abord souligner.

Une fonction possède toujours trois éléments que tu dois écrire avant le crochet ouvrant (voir la figure 4-2).

Le premier élément est le type de la donnée qui sera retournée par la fonction. Pour la plupart des fonctions que tu vas créer dans ce livre, il s'agira de `void`, ce qui signifie qu'aucune information ne sera fournie en retour. Si une donnée est retournée par la fonction, son type doit être indiqué à la place de `void`. Par exemple, si ta fonction calcule un résultat qui est un entier, le type de la donnée retournée doit être `int` au lieu de `void`.

Copyright © 2015 Eyrolles.

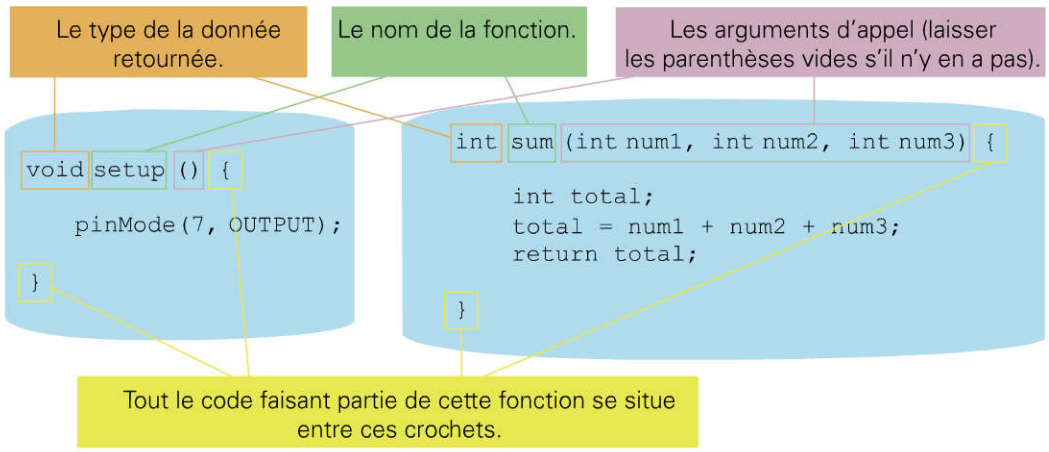


FIGURE 4-2 Anatomie d'une fonction.

Tu as peut-être remarqué que le terme `void` apparaît avant `setup()` et `loop()`. Cela signifie que ces deux fonctions ne renvoient aucune information quand elles ont terminé leur exécution. En revanche, tu peux, par exemple, écrire une fonction qui calcule la somme de trois nombres et qui en renvoie le résultat en le plaçant là où tu l'appelles afin de pouvoir mémoriser cette valeur dans une variable. Cette variable pourrait être un entier `int`. Pour les fonctions qui ne font qu'allumer ou éteindre des lampes, aucune information n'a besoin d'être renvoyée, leur type de retour est donc `void`.

Le second élément est le nom de la fonction. Les règles pour nommer les fonctions sont identiques à celles des noms de variables qui sont décrites dans l'aventure 2.

- Tu ne peux pas utiliser d'espaces dans le nom, seulement des lettres et des chiffres.
- Tu ne peux pas commencer un nom de fonction par un chiffre.
- La première lettre doit être une minuscule.
- Tu ne dois pas utiliser de lettres accentuées.

Tu ne verras peut-être pas facilement le dernier élément constitutif d'une fonction : il s'agit des parenthèses. Quand rien n'est présent entre la première parenthèse `(` et la seconde `)`, cela signifie qu'aucun argument n'est fourni en entrée à la fonction. Tu peux lui transmettre des informations en utilisant des arguments d'entrée, tu verras comment faire plus loin.

Reviens au sketch et place ton curseur sur la dernière ligne du code, celle qui est située après le crochet de fermeture de la fonction `loop()`. Ajoute le code suivant, en faisant attention qu'il ne soit pas situé dans une autre fonction (en d'autres termes, vérifie qu'il n'est pas situé entre les parenthèses ou crochets d'une autre fonction).

Copyright © 2015 Eyrolles.

Le code original :

```
// Turn on the LED for 1 second
// then off for 1 second
void blinkOnce() {
  digitalWrite(led, HIGH); // turn the LED on
  delay(1000); // wait for a second
  digitalWrite(led, LOW); // turn the LED off
  delay(1000); // wait for a second
}
```

Le code francisé :

```
// Allumage de la LED pendant une seconde
// puis extinction pendant une seconde
void clignotementUnique() {
  digitalWrite(led, HIGH); // allume la LED
  delay(1000); // pause d'une seconde
  digitalWrite(led, LOW); // éteint la LED
  delay(1000); // pause d'une seconde
}
```

Tu viens de créer une nouvelle fonction nommée `blinkOnce()` ou `clignotementUnique()`. Elle ne reçoit aucun argument en entrée (car aucun argument n'est présent entre les parenthèses `( )` placées après son nom) et ne retourne aucune valeur (car le type `void` est indiqué avant son nom). À l'intérieur de la fonction, les instructions allument puis éteignent la LED pendant une seconde.

L'ajout de commentaires au début de chaque fonction est un excellent principe, cela permet d'expliquer le rôle de la fonction et ce qu'elle fait. Tu peux utiliser `//` au début de chaque ligne de commentaire ou `/*` et `*/` au début et à la fin d'un paragraphe de commentaires. Cela te paraît peut-être inutile quand il s'agit d'une fonction simple, mais si tu le fais systématiquement, tu te rappelleras toujours qu'il faut ajouter des commentaires pour les fonctions plus complexes. Et quand tu partageras du code bien commenté, les autres pourront le comprendre plus facilement.

Modifie maintenant la fonction `loop()` d'origine afin qu'elle ne fasse plus qu'appeler ta nouvelle fonction :

```
// The loop routine runs over and over again forever :
void loop() {
  blinkOnce();
}
```

Copyright © 2015 Eyrolles.

Ou, pour la version francisée :

```
// La fonction loop s'exécute indéfiniment :
void loop() {
  clignotementUnique();
}
```

Téléverse le croquis dans ta carte Arduino Uno après l'avoir connectée en cliquant sur le bouton Téléverser. Tu devrais voir la LED clignoter exactement comme le faisait le sketch `Blink` original de l'aventure 1.

Tu vas maintenant découvrir ce dont les fonctions sont capables. Tu vas écrire un peu plus de code pour ajouter une variable qui contrôlera la vitesse du clignotement.

Change ta fonction `blinkOnce()` ou `clignotementUnique()` de la façon suivante (les changements sont en gras) :

Le code original :

```
// Turn on the LED for time passed in argument
// then off for time passed in argument
void blinkOnce(int time) {
  digitalWrite(led, HIGH); // turn the LED on
  delay(time); // wait for a second
  digitalWrite(led, LOW); // turn the LED off
  delay(time); // wait for a second
}
```

Le code francisé :

```
// Allumage de la LED pendant le temps passé en argument
// puis extinction pendant le temps passé en argument
void clignotementUnique(int temps) {
  digitalWrite(led, HIGH); // allume la LED
  delay(temps); // pause d'une seconde
  digitalWrite(led, LOW); // éteint la LED
  delay(temps); // pause d'une seconde
}
```

En ajoutant une variable entre les parenthèses `(` et `)`, ton programme indique à la fonction qu'elle a besoin de plus d'informations pour s'exécuter. On appelle cela un argument grâce auquel tu peux passer une information directement d'une fonction à une autre.

Cela signifie que tu dois aussi ajouter l'argument lors de l'appel à la fonction `blinkOnce()` ou `clignotementUnique()` dans la fonction `loop()`. Voici comment faire :

Copyright © 2015 Eyrolles.

```
// The loop routine runs over and over again forever :
void loop() {
  blinkOnce(1000);
}
```

Ou, pour la version francisée :

```
// La fonction loop s'exécute indéfiniment :
void loop() {
  clignotementUnique(1000);
}
```

En ajoutant l'argument `1000`, la LED clignotera exactement comme elle le faisait avant : allumée pendant 1 seconde puis éteinte pendant 1 seconde.

Utilisation des boucles for

Une autre méthode pour organiser ton code consiste à répéter une action un certain nombre de fois. Par exemple, au lieu de prendre le risque de commettre des erreurs en écrivant cinq fois la même instruction à la suite, tu peux utiliser une autre méthode qui le fera pour toi. La boucle `for` permet de le faire.

Une boucle `for` a besoin de trois informations essentielles, comme le montre la figure 4-3.

La première information décrit la condition de départ. Une boucle `for` débute avec une valeur de départ mémorisée dans une variable. C'est en général une variable temporaire qui n'est utilisée que dans la boucle. Tu peux la nommer comme tu le veux, mais les programmeurs ont tendance à l'appeler tout simplement `i`.

DÉFI

Fais clignoter ta LED pendant 1 seconde puis pendant 5 secondes. Tu vas devoir appeler la fonction `blinkOnce()` ou `clignotementUnique()` deux fois dans la fonction `loop()` en utilisant différents arguments.

Copyright © 2015 Eyrolles.

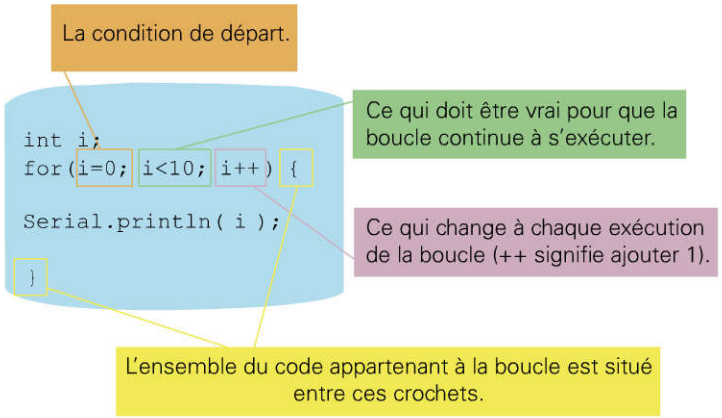


FIGURE 4-3 Anatomie d'une boucle for.

La seconde information est ce qui est nécessaire à la boucle pour qu'elle s'arrête. Elle s'énonce sous la forme d'une question dont la réponse est `true` ou `false` (vrai ou faux). Souvent, elle consiste à vérifier la valeur d'une variable qui ne doit pas dépasser une limite, par exemple `i<10`.

La dernière information est ce qui arrive à cette variable `i` après chaque exécution de la boucle. La variable doit évoluer depuis sa valeur de départ jusqu'à la condition provoquant l'arrêt de la boucle. Sinon, la boucle continuerait indéfiniment et ton programme n'atteindrait jamais le code placé à la suite de la boucle `for`. Cette partie de code est souvent écrite `i++`, ce qui signifie simplement ajouter `1` à la variable `i` et mémoriser cette nouvelle valeur dans `i`.

Crée un nouveau sketch (soit en cliquant sur le bouton Nouveau de l'IDE Arduino, soit en utilisant le menu Fichier > Nouveau). Saisis le code suivant.

La version originale :

```
void setup() {
  Serial.begin(9600);
}
void loop() {
  int i;
  for(i=0; i<10; i++){ // for loop that counts from 0 to 9
    Serial.println(i); // print the current value of i
    delay(1000); // wait for 1 second
  }
  delay(3000); // wait for 3 seconds
}
```

La version francisée :

```
void setup() {
  Serial.begin(9600);
```

Copyright © 2015 Eyrolles.

```
}
void loop() {
  int i;
  for(i=0; i<10; i++){ // boucle for comptant de 0 à 9
    Serial.println(i); // affiche la valeur de i
    delay(1000); // pause d'une seconde
  }
delay(3000); // pause de trois secondes
}
```

Télécharge ce programme dans ta carte Arduino et ouvre le moniteur série de l'IDE Arduino en cliquant sur le bouton Moniteur série ou à partir du menu Outils > Moniteur série. Tu devrais voir la boucle `for` compter de 0 à 9 indéfiniment.

Essaie de remplacer `i++` par `i+=2`. Que va-t-il se passer à ton avis ?

Comment obtenir plus de sorties avec un registre à décalage ?

Dans les précédentes aventures de ce livre, tu as construit des circuits avec des composants électriques essentiels. Les résistances sont des composants de base que tu peux combiner avec d'autres composants élémentaires pour réaliser des circuits plus complexes. Cependant, il est parfois inutile de passer du temps (et d'utiliser beaucoup d'espace) pour construire un circuit compliqué. En effet, quelques fois, un composant contenant des circuits plus complexes a déjà été réalisé pour toi. Ces composants sont appelés **circuits intégrés**, ou abréviés IC (*Integrated Circuits* en anglais).

Les **circuits intégrés** (IC) sont des circuits contenus dans une seule puce. Le même circuit peut exister dans différentes formes que l'on appelle des boîtiers. Quand tu travailles avec une plaque de montage, tu dois utiliser des circuits dont le boîtier est de type DIP ou DIL (*Dual Inline Package* en anglais, ou boîtier à deux rangées de broches alignées). Ces boîtiers ont des broches qui s'insèrent dans les plaques de montage.

Copyright © 2015 Eyrolles.

Les circuits intégrés existent dans différentes formes de boîtiers. Cela signifie simplement qu'ils ont des formes et des dimensions variées. Quand tu utilises des plaques de montage, tu dois choisir des circuits intégrés de type **DIP** ou **DIL** *(Dual Inline Package)*, car ils ont des broches prévues pour s'insérer dans les plaques de montage. L'autre type de boîtier est le **CMS** (**Composant monté en surface**), appelé en anglais *Surface Mounted Device* ou SMD). Les composants CMS sont très petits et sont prévus pour être facilement placés sur les cartes de circuits imprimés lors de leur fabrication. Ils sont beaucoup plus difficiles à utiliser pour la réalisation de circuits à l'aide de plaques de montage. La plupart des composants de ta carte Arduino Uno (tous ceux qui sont noirs ainsi que les LED) sont des composants CMS.

Un boîtier **Dual Inline Package** (DIP ou DIL) est un de format de circuits intégrés. Il possède deux rangées de broches qui peuvent s'insérer dans les rangées d'une plaque de montage.

Un **Composant monté en surface** (CMS) est un format de circuits intégrés ou d'autres composants comme les résistances. Il est réalisé de telle sorte qu'il puisse être soudé sur une surface plane sans que ses broches ne traversent les trous du circuit imprimé.

Tu peux utiliser plusieurs composants dans un même circuit pour répéter une même fonction de façon répétitive. Tu peux considérer les circuits intégrés comme étant les composants électroniques qui procurent des fonctions.

Dans cette aventure, tu vas utiliser un circuit intégré appelé **registre à décalage**. Celui dont tu vas te servir possède trois entrées, qui vont te permettre de contrôler ce qui se passe sur huit sorties. Ainsi, avec seulement trois sorties de ta carte Arduino Uno, tu peux contrôler huit LED différentes. Mieux encore, tu peux relier en chaîne deux registres à décalage. De cette façon, tu peux ajouter huit LED supplémentaires tout en utilisant uniquement les trois broches de sortie de ta carte Arduino !

Copyright © 2015 Eyrolles.

Un **registre à décalage** est un composant qui contrôle plusieurs sorties en utilisant un nombre réduit de signaux d'entrée. C'est un composant fréquemment utilisé pour contrôler un grand nombre de LED.

Comment fonctionne un registre à décalage ?

Les trois entrées utilisées par un registre à décalage sont `CLOCK` (l'horloge), `DATA` (les données) et `LATCH` (le verrouillage).

Clock

L'entrée `CLOCK` est le tambour battant du circuit. Les messages sont envoyés depuis la carte Arduino vers le circuit intégré. Tu peux comparer Arduino à un chanteur chantant en chœur avec le circuit intégré. Afin qu'il puisse suivre, Arduino et lui ont besoin de chanter au même rythme. L'entrée `CLOCK` est une suite de valeurs `HIGH` et `LOW` (niveaux HAUT et BAS) qui informent le circuit intégré quand une nouvelle information est transmise, comme un battement de tambour permettant au circuit intégré de suivre Arduino (voir la figure 4-4).

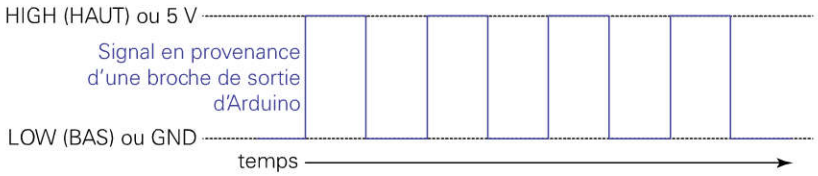


FIGURE 4-4 Le signal CLOCK (horloge).

Data

Le signal `DATA` est la valeur que tu souhaites que chaque sortie (LED) prenne et qui sera `HIGH` ou `LOW` (HAUT ou BAS). Un registre à décalage peut contrôler jusqu'à huit LED. La carte Arduino Uno envoie une par une au registre à décalage les valeurs à appliquer aux LED, de la façon suivante.

1. La valeur de la première LED est envoyée par l'Arduino Uno au registre à décalage, qui l'applique alors à sa première broche de sortie.
2. La valeur de la seconde LED est envoyée au registre à décalage qui applique à la seconde sortie la valeur qui était mémorisée dans la première sortie puis applique la dernière valeur reçue de la Uno à sa première sortie.
3. L'Arduino Uno envoie au registre à décalage la valeur de la troisième LED. La valeur de la troisième sortie devient alors celle mémorisée précédemment dans la seconde

Copyright © 2015 Eyrolles.

sortie, la seconde sortie prend la valeur mémorisée précédemment dans la première sortie et la première sortie prend la dernière valeur reçue de l'Arduino Uno.

4. La carte Arduino Uno continue ainsi à envoyer de nouvelles valeurs au registre à décalage. Chaque fois qu'il reçoit une nouvelle valeur, le registre à décalage déplace toutes les valeurs précédemment reçues pour chaque broche à la broche suivante : la première sortie reçoit donc toujours la valeur la plus récente.

Chaque fois que tu envoies une nouvelle valeur au registre à décalage, la valeur précédente est décalée à la sortie suivante. D'où le nom de « registre à décalage » !

Quand tu as terminé d'envoyer tes données au registre à décalage, tu dois lui dire que tu as fini afin qu'il puisse établir les niveaux HAUT et BAS sur chaque broche de sortie : c'est ce que permet l'entrée `LATCH`.

Latch

Le dernier signal d'entrée indique au registre à décalage soit d'attendre d'autres données, soit de poursuivre en envoyant effectivement les niveaux qu'il a mémorisés à ses broches de sortie. Quand la broche `LATCH` est au niveau bas (`LOW`), le circuit intégré est en attente ; quand elle est au niveau haut (`HIGH`), le registre à décalage applique les valeurs mémorisées sur ses broches de sortie. Si des LED sont reliées au registre à décalage, elles seront allumées ou éteintes en fonction des nouvelles valeurs mémorisées dans le registre à décalage quand son entrée `LATCH` passe au niveau `HIGH`.

La figure 4-5 montre comment fonctionnent les trois entrées de contrôle du registre à décalage.

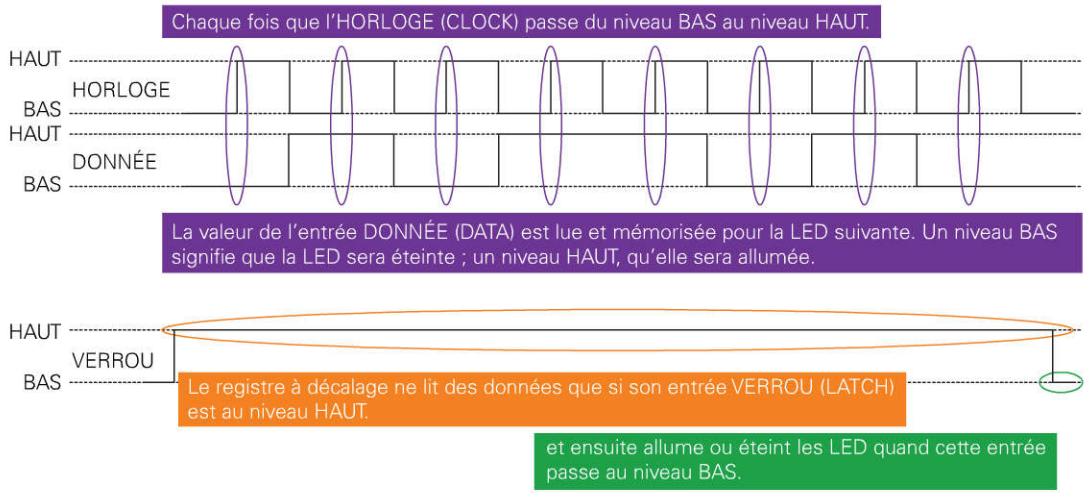


FIGURE 4-5 Le fonctionnement du registre à décalage.

Copyright © 2015 Eyrolles.

Réalisation des connexions de ton registre à décalage

En premier lieu, tu dois réaliser ton circuit. Commence par insérer ton registre à décalage sur ta plaque de montage. Le circuit intégré sera placé au-dessus de la séparation entre les deux groupes de rangées, au milieu de la plaque. Il sera peut-être nécessaire que tu plies légèrement les broches du composant pour qu'elles s'insèrent correctement dans les trous. Tu remarqueras qu'un petit point est gravé ou imprimé dans un coin du circuit intégré ou qu'un demi-cercle apparaît sur un de ses petits côtés : c'est le sommet du circuit intégré. Il est très important qu'il ait la même orientation que sur le plan de la figure 4-6.

FIGURE 4-6 Le diagramme de connexion du registre à décalage.

Les broches d'un circuit intégré sont fragiles, fais très attention si tu dois les plier. Sois aussi très méticuleux en retirant un circuit intégré de ta plaque de montage, ses broches peuvent facilement se plier durant cette opération.

Te voici prêt à construire le circuit de ton registre à décalage. Le brochage et le plan des connexions à réaliser sont représentés sur la figure 4-6, qui montre le composant et indique ses broches et ce qui doit être relié à chacune d'elles. Tu remarqueras que les couleurs correspondent aux couleurs des fils de la figure 4-7. Tu dois maintenant réaliser la première liaison selon les étapes suivantes (ne connecte pas encore ta carte Arduino Uno à ton ordinateur).

1. Utilise un fil de pontage pour relier une des grandes rangées en bas de ta plaque de montage à la broche GND de la carte Arduino Uno. Si ta carte possède un marquage identifiant la ligne GND (ligne bleue, noire ou -), utilise cette rangée.
2. Utilise un autre fil de pontage pour relier l'autre grande rangée en bas de ta plaque à la broche +5 V de ta carte Arduino Uno.
3. Utilise deux fils de pontage pour relier chacune des grandes rangées en bas de ta plaque de montage aux grandes rangées situées en haut de ta plaque. Si ta plaque

Copyright © 2015 Eyrolles.

comporte un marquage, relie la rangée identifiée par une ligne rouge ou le signe + à la rangée du haut qui possède le même marquage, puis relie les deux autres rangées ensemble.

4. Utilise deux fils de pontage pour rattacher les petites rangées reliées aux broches 8 et 13 du registre à décalage à la grande rangée reliée à la masse GND.
5. Utilise deux fils de pontage pour rattacher les petites rangées rattachées aux broches 10 et 16 du registre à décalage à la grande rangée reliée à l'alimentation +5 V.
6. Utilise un fil de pontage pour rattacher la petite rangée reliée à la broche 14 du registre à décalage à la broche 11 de ta carte Arduino Uno.
7. Utilise un fil de pontage pour rattacher la petite rangée reliée à la broche 11 du registre à décalage à la broche 12 de ta carte.
8. Utilise encore un fil de pontage pour rattacher la petite rangée reliée à la broche 12 du registre à décalage à la broche 8 de ta carte.

Quand tu auras terminé, ton circuit devra ressembler à celui de la figure 4-7. Assure-toi que le registre à décalage au milieu de la plaque est correctement orienté.

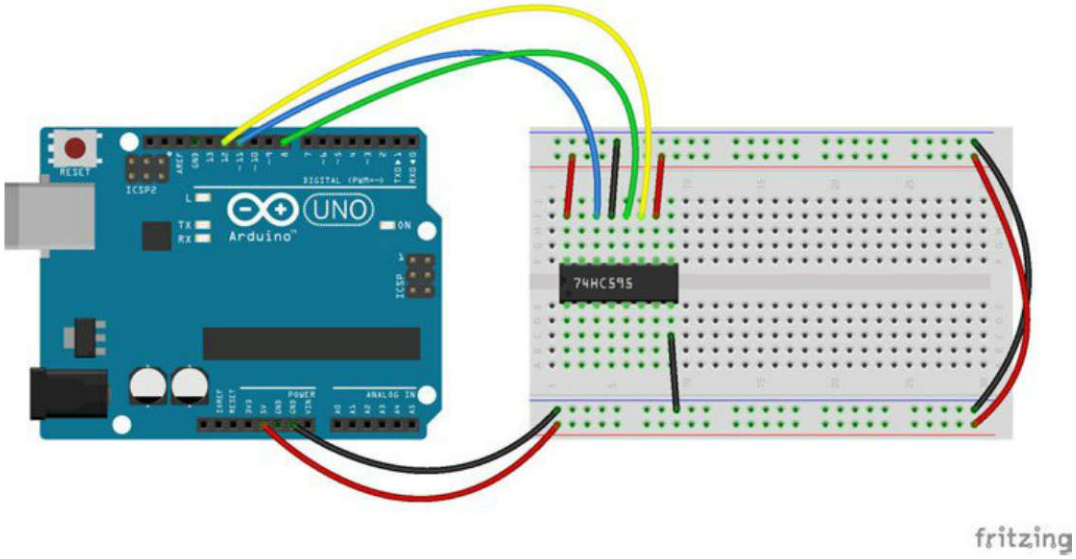


FIGURE 4-7 Les premières connexions du registre à décalage.

Ajout de LED

Il est maintenant temps d'ajouter les LED. Si tu as une deuxième plaque de montage, il sera plus facile de disposer le registre à décalage sur une plaque et les LED sur l'autre, mais tu peux aussi tout relier sur la même plaque.

Chacune des LED à besoin d'une résistance de limitation de courant, comme lorsqu'une LED est reliée à une sortie de ta carte Arduino Uno. Chaque sortie du registre à décalage

Copyright © 2015 Eyrolles.

est donc reliée à une résistance de limitation de courant puis à une LED qui est ensuite reliée à la masse GND.

En te guidant à l'aide de la figure 4-8, suis ces étapes pour ajouter tes LED et leurs résistances.

1. Insère les fils les plus courts des 8 LED dans la grande rangée reliée à la masse GND, en haut de la plaque.
2. Relie le fil le plus long de chaque LED dans sa propre rangée courte de la plaque de montage, à n'importe quel endroit qui te convient (aucune rangée courte n'étant encore utilisée).
3. Insère un des fils d'une résistance dans chaque rangée recevant déjà le fil le plus long de chaque LED.
4. Plie les résistances afin qu'elles se situent au-dessus de la séparation centrale des rangées de la plaque. Insère les fils libres des résistances dans la rangée courte en dessous de la séparation centrale.
5. Utilise huit fils de pontage pour relier chaque résistance à une des broches de sortie du registre à décalage : les broches 15 et 1 à 7. Guide-toi en regardant la figure 4-8.

Vérifie soigneusement que tes connexions sont correctement effectuées en suivant à nouveau chaque étape. Quand tu es sûr que tout est au bon emplacement, tu es prêt à mettre ton circuit sous tension en reliant ta carte Arduino Uno à ton ordinateur.

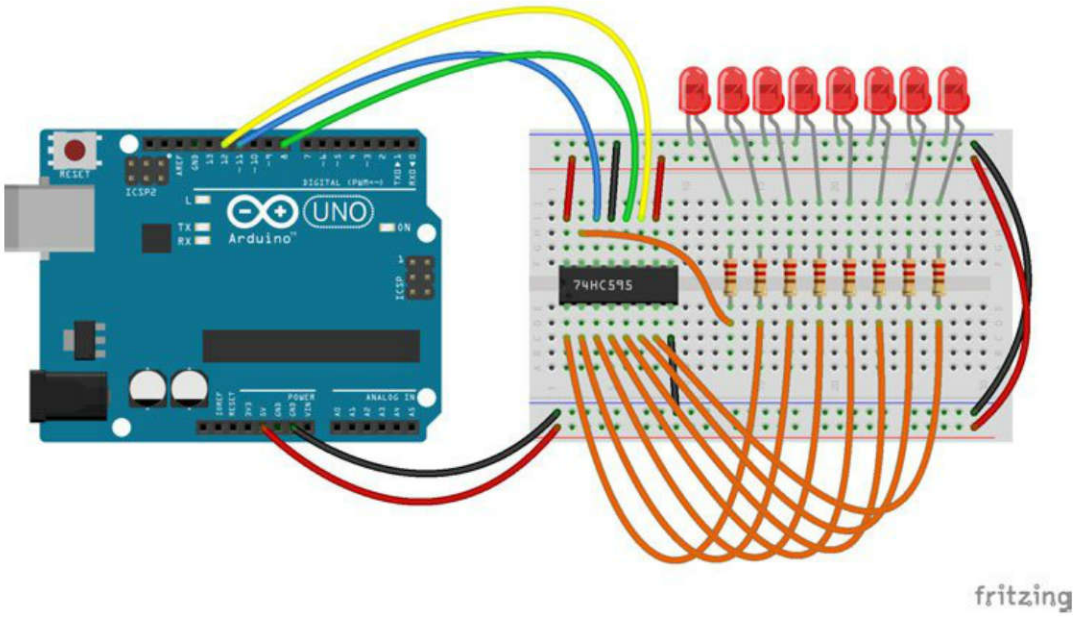


FIGURE 4-8 Le circuit du registre à décalage complété.

Copyright © 2015 Eyrolles.

Ton circuit intégré ne doit **jamais** devenir chaud, car cela signifie que quelque chose n'est pas bon et que tu dois tout de suite déconnecter l'alimentation. Sinon, le circuit intégré pourrait être détruit et il te faudrait alors le remplacer. Si tu entends un petit claquement sec ou s'il fume, il devra absolument être remplacé. Dans ce cas, vérifie à nouveau ton circuit à la recherche d'erreurs, avant de le reconnecter à ton circuit intégré pour ne pas risquer de détruire le nouveau composant de la même façon !

Écriture du programme

Crée un nouveau sketch en cliquant sur le bouton Nouveau de l'IDE Arduino ou en utilisant le menu Fichier > Nouveau. Saisis le code suivant :

```
void setup() {
}
void loop() {
}
```

Tout en haut de ton sketch, avant l'instruction `setup()`, ajoute les variables suivantes :

La version originale :

```
// The pin connected to latch pin (ST_CP) of 74HC595
int latchPin = 8;
// The pin connected to clock pin (SH_CP) of 74HC595
int clockPin = 12;
// The pin connected to Data in (DS) of 74HC595
int dataPin = 11;
```

La version francisée :

```
// La broche reliée à l'entrée LATCH (verrou) du 74HC595
int brocheVerrou = 8;
// La broche reliée à l'entrée CLOCK (horloge) du 74HC595
int brocheHorloge = 12;
// La broche reliée à l'entrée DATA (données) du 74HC595
int brocheDonnees = 11;
```

Trois broches, respectivement `LATCH` (verrou), `CLOCK` (horloge) et `DATA` (données) sont prévues pour relier le registre à décalage.

Saisis ensuite les instructions suivantes entre les crochets `{ }` de la fonction `setup()` :

Copyright © 2015 Eyrolles.

La version originale :

```
// Set pins to output because they are addressed in the main
// loop
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
```

La version francisée :

```
// Déclaration en mode sortie des broches utilisées dans la
// boucle principale
pinMode(brocheVerrou, OUTPUT);
pinMode(brocheDonnees, OUTPUT);
pinMode(brocheHorloge, OUTPUT);
```

Pour terminer, entre les lignes de code suivantes entre les crochets { } de la fonction `loop()` :

La version originale :

```
// loop through 0 to 256
int i;
for(i=0; i<256; i++) {
  // turn off the output so the pins don't light up
  // while you're shifting bits
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, LSBFIRST, i);

  // turn on the output so the LEDs can light up
  digitalWrite(latchPin, HIGH);
  delay(300);
}
```

La version francisée :

```
// Boucle de 0 à 256
int i;
for(i=0; i<256; i++) {
  // Verrouillage au niveau bas des sorties du registre à
  // décalage afin que les LED soient éteintes lors de
  // l'envoi des données
  digitalWrite(brocheVerrou, LOW);
  shiftOut(brocheDonnees,brocheHorloge, LSBFIRST, i);
```

Copyright © 2015 Eyrolles.

```
  // Déverrouillage des sorties du registre
  // afin que les LED s'allument
  digitalWrite(brocheVerrou, HIGH);
  delay(300);
}
```

Ton sketch complété devrait ressembler à ceci :

La version originale :

```
// The pin connected to latch pin (ST_CP) of 74HC595
int latchPin = 8;
// The pin connected to clock pin (SH_CP) of 74HC595
int clockPin = 12;
// The pin connected to Data in (DS) of 74HC595
int dataPin = 11;

void setup() {
  // Set pins to output because they are addressed in the
  // main loop
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
}

void loop() {
  // loop through 0 to 256
  int i;
  for(i=0; i<256; i++) {
    // turn off the output so the pins don't light up
    // while you're shifting bits
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, i);
    // turn on the output so the LEDs can light up
    digitalWrite(latchPin, HIGH);
    delay(300);
  }
}
```

La version francisée :

```
// La broche reliée à l'entrée LATCH (verrou) du 74HC595
int brocheVerrou = 8;
// La broche reliée à l'entrée CLOCK (horloge) du 74HC595
int brocheHorloge = 12;
// La broche reliée à l'entrée DATA (données) du 74HC595
int brocheDonnees = 11;
```

Copyright © 2015 Eyrolles.

```
void setup() {
  // Déclaration en mode sortie des broches utilisées dans
  // la boucle principale
  pinMode(brocheVerrou, OUTPUT);
  pinMode(brocheDonnees, OUTPUT);
  pinMode(brocheHorloge, OUTPUT);
}

void loop() {
  // Boucle de 0 à 256
  int i;
  for(i=0; i<256; i++) {
    // Verrouillage au niveau bas des sorties du registre à
    // décalage afin que les LED soient éteintes lors de
    // l'envoi des données
    digitalWrite(brocheVerrou, LOW);
    shiftOut(brocheDonnees,brocheHorloge, LSBFIRST, i);

    // Déverrouillage des sorties du registre
    // afin que les LED s'allument :
    digitalWrite(brocheVerrou, HIGH);
    delay(300);
  }
}
```

Tu peux télécharger les sketches de cette aventure depuis le site d'accompagnement (www.editions-eyrolles.com/go/arduino).

Télécharge le sketch dans ta carte Arduino Uno. Tes LED devraient alors commencer à clignoter !

EXPLORATION DU CODE

Que se passe-t-il dans la fonction `loop()` ? Elle emploie une boucle `for`, décrite précédemment dans cette aventure. À l'intérieur de cette boucle `for`, tu appelles une fonction nommée `shiftOut()` : c'est une fonction connue d'Arduino qui reçoit quatre arguments. Le premier est le numéro de la broche sur laquelle les données sont envoyées (`dataPin` ou `brocheDonnees`) en

Copyright © 2015 Eyrolles.

utilisant la broche désignée dans le second argument (`clockPin` ou `brocheHorloge`) comme signal `CLOCK`. Le troisième argument utilise un mot-clé pour indiquer si la donnée envoyée correspond à la première position du registre ou à sa dernière position (`LSBFIRST` signifie que le bit le moins significatif est en premier). Le dernier argument est la valeur de la donnée transmise. Ici, la valeur envoyée est mémorisée dans la variable `i`, qui est contrôlée par la boucle `for`. Elle débute à la valeur `i=1` et s'arrête lorsque `i=255`.

```
shiftOut(dataPin, clockPin, LSBFIRST, i) ;
```

ou, dans la version francisée :

```
shiftOut (brocheDonnees, brocheHorloge, LSBFIRST, i) ;
```

Avant que la fonction `shiftOut()` ne soit appelée, la broche `latchPin` (ou `brocheVerrou`) est positionnée au niveau bas (`LOW`). Cela indique au registre à décalage de stopper son fonctionnement et de se mettre en veille, à l'écoute. Les nouvelles valeurs du registre sont alors transmises par la fonction `shiftOut()` puis la broche `latchPin` (ou `brocheVerrou`) est mise au niveau haut (`HIGH`). Cela indique au registre à décalage de cesser d'être à l'écoute et de continuer. Il éteint ou allume alors les LED selon les nouvelles valeurs qu'il vient de recevoir.

Tu as peut-être noté quelque chose d'étrange dans le code. Pourquoi la boucle `for` démarre à 0 et compte jusqu'à 256 ? Cela n'est-il pas un peu bizarre ?

Les ordinateurs aiment commencer à compter à partir de 0. En effet, alors que nous, humains, sautons la valeur 0 et commençons à compter à partir de 1, les ordinateurs commencent en général par 0. C'est pourquoi la boucle `for` commence à 0.

Pourquoi utiliser le nombre `255` comme valeur maximale ? Le registre à décalage contrôle huit LED. Chaque LED peut être soit allumée, soit éteinte, c'est donc deux états possibles pour chaque LED. Si 0 représente une LED éteinte et 1 une LED allumée, tu peux représenter les états de toutes les LED à l'aide d'un nombre unique, 11111111 pour toutes les LED allumées, 10000001 représentant toutes les LED éteintes à l'exception de la première et de la dernière.

Ces nombres sont spéciaux en cela qu'ils n'utilisent pas tous les chiffres possibles entre 0 et 9, mais seulement 0 et 1. Les nombres qui utilisent les valeurs 0 à 9 sont appelés des nombres décimaux (ceux que tu considères comme des nombres normaux) et les nombres qui utilisent uniquement les valeurs 0 et 1 sont appelés des nombres binaires. Les nombres transmis par la sortie `dataPin` (ou `brocheDonnees`) sont représentés par des valeurs binaires.

Copyright © 2015 Eyrolles.

Le sketch que tu viens d'écrire est un compteur binaire : il te montre, à l'aide des LED, comment compter de 0 à 255 en binaire (de 0 à 11111111).

Un **nombre binaire** utilise uniquement les chiffres 0 et 1, contrairement aux nombres décimaux qui utilisent les chiffres 0 à 9. Le système binaire est aussi appelé système en base 2. Le système décimal étant en base 10.

La figure 4-9 te montre comment convertir un nombre binaire en nombre décimal. À ce stade, tu n'as pas à te soucier de cela, les différentes façons de représenter un nombre n'étant pas ce qu'il y a de plus amusant. Par contre, si tu aimes les messages secrets codés, c'est un sujet qui pourrait t'intéresser.

Pour calculer la valeur décimale que représente une combinaison de LED allumées ou éteintes :

Multiplier la valeur 0 ou 1 par 2 élevé à la puissance correspondant à la position de la lampe, selon qu'elle est allumée ou éteinte.

Puis additionner tous les résultats de ces multiplications.

```
128 x 1
 64 x 0
 32 x 1
 16 x 0
  8 x 1
  4 x 0
  2 x 1
+ 1 x 0
  170
```

1 pour les lampes allumées,
0 pour les lampes éteintes.

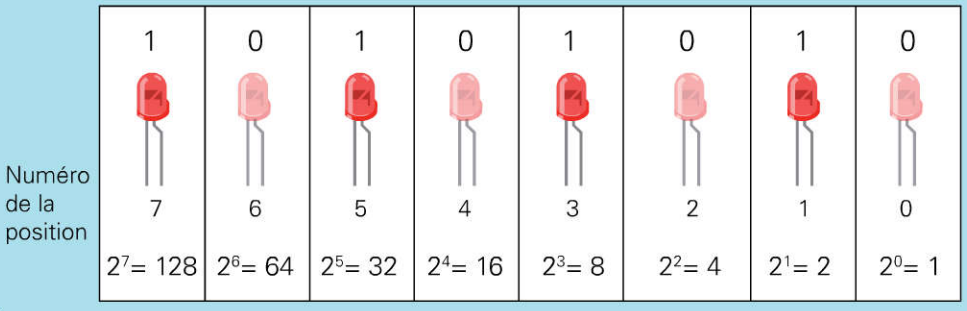


FIGURE 4-9 Comment convertir un nombre binaire en nombre décimal.

Copyright © 2015 Eyrolles.

DÉFI

Calcule la valeur décimale de la combinaison binaire de la figure 4-10.

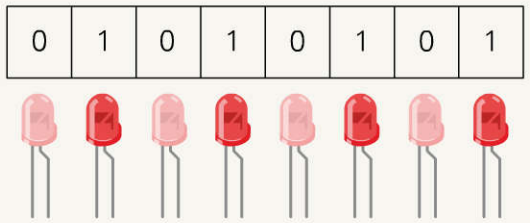


FIGURE 4-10 Quel nombre décimal représenterait cette combinaison binaire ?

Un chiffre d'un nombre binaire est appelé « bit » et un groupe de 8 bits est appelé un « octet » (*byte*, en anglais). Tous les calculateurs utilisent des bits et des octets *(bytes)*. Si cela t'intéresse d'en apprendre davantage, tu peux trouver plus d'informations sur les bits et les octets sur les sites en français `http://fr.wikipedia.org/wiki/Bit` et `http://fr.wikipedia.org/wiki/Byte`.

Ajout d'autres registres à décalage

Pour ajouter un autre registre à décalage, tu dois placer un second circuit intégré sur ta plaque de montage (si tu utilises une plaque de grande dimension) ou sur une seconde plaque (si tu utilises deux petites plaques).

1. Suis les étapes précédentes pour relier le registre à décalage à l'alimentation +5 V et à la masse GND. Si tu utilises une deuxième plaque de montage, assure-toi de bien relier les grandes rangées de la première plaque à celles de la seconde.
2. Au lieu de relier les broches 12, 14 et 11 du registre à décalage aux broches 8, 11 et 12 de ta carte Arduino Uno, relie les broches 11 et 12 du deuxième registre à décalage aux broches 11 et 12 du premier. Cela relie les entrées `LATCH` (verrou) et `CLOCK` (horloge) des deux registres à décalage ensemble, comme indiqué sur la figure 4-11.
3. L'entrée `DATA` (données) du deuxième registre à décalage ne vient pas de la carte Arduino Uno, mais de la broche 9 du premier registre à décalage. Utilise un fil de pontage pour relier la broche 14 du deuxième registre à décalage à la broche 9 du premier.
4. Pour connecter 8 LED supplémentaires au second registre à décalage, suis les étapes de la section « Ajout de LED » page 135, décrivant l'ajout des LED du premier registre à décalage.

Copyright © 2015 Eyrolles.

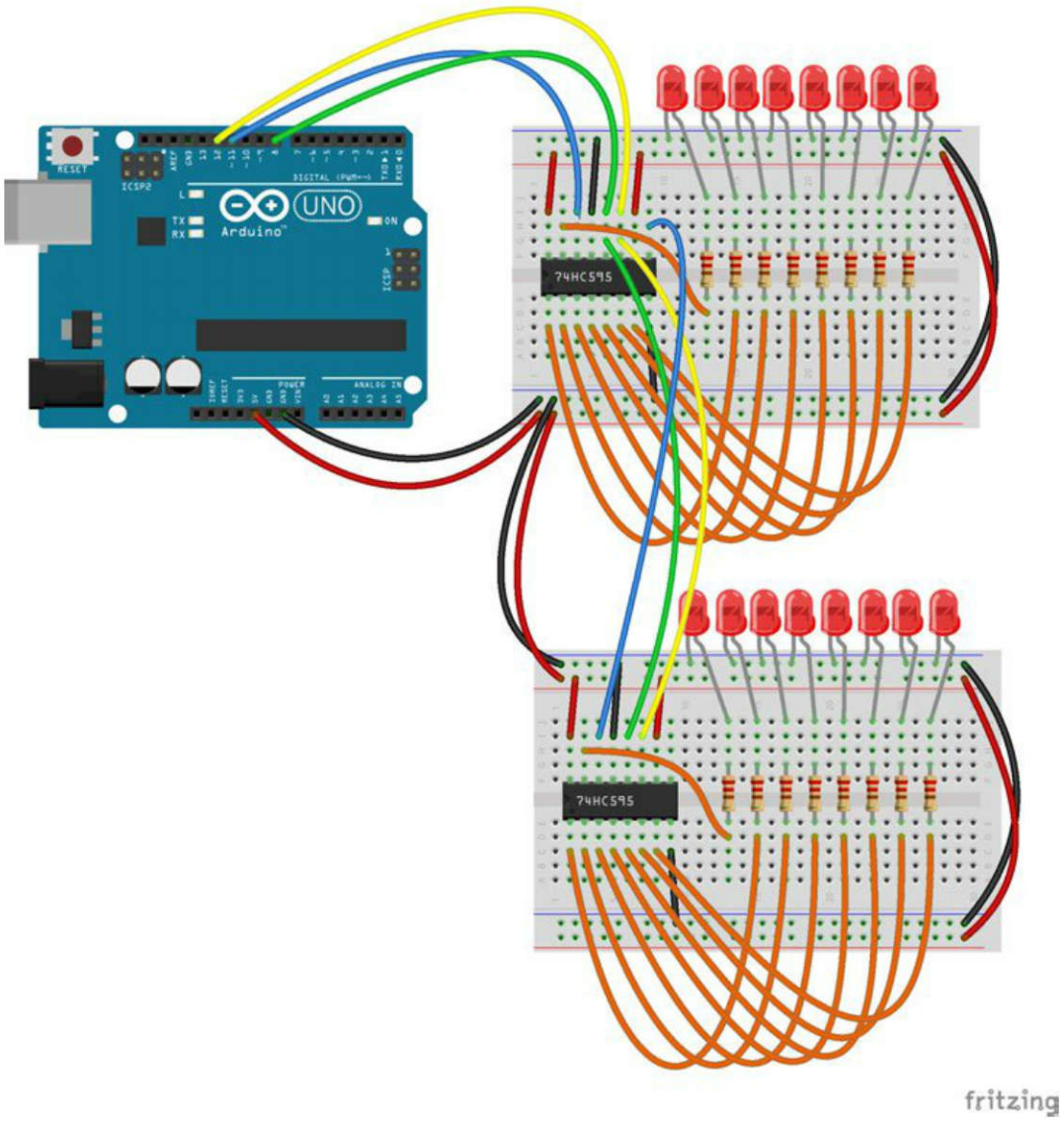


FIGURE 4-11 Ajout d'un second registre à décalage.

Quand tu auras réalisé ton circuit, change ton sketch de façon à ce qu'il contrôle deux registres à décalage au lieu d'un seul. Fais les modifications indiquées en gras dans la fonction `loop()` de ton sketch.

La version originale :

```
void loop() {
  // loop through 0 to 256
  int i;
  for(i=0; i<256; i++) {
    // turn off the output so the pins don't light up
    // while you're shifting bits
    digitalWrite(latchPin, LOW);
```

Copyright © 2015 Eyrolles.

```
    // send to second shift register
    shiftOut(dataPin, clockPin, LSBFIRST, i);
    // send to first shift register
    shiftOut(dataPin, clockPin, LSBFIRST, i);

    // turn on the output so the LEDs can light up
    digitalWrite(latchPin, HIGH);
    delay(300);
  }
}
```

La version francisée :

```
void loop() {
  // Boucle de 0 à 256
  int i;
  for(i=0; i<256; i++) {
    // Verrouillage au niveau bas des sorties du registre à
    // décalage afin que les LED soient éteintes lors de
    // l'envoi des données
    digitalWrite(brocheVerrou, LOW);

    // Envoi des données au second registre à décalage
    shiftOut(brocheDonnees, brocheHorloge, LSBFIRST, i) ;
    // Envoi des données au premier registre à décalage
    shiftOut(brocheDonnees,brocheHorloge, LSBFIRST, i);

    // Déverrouillage des sorties du registre
    // afin que les LED s'allument
    digitalWrite(brocheVerrou, HIGH);
    delay(300);
  }
}
```

Télécharge ton sketch dans ta carte Arduino Uno, et contemple les seize LED s'allumer et s'éteindre.

Réalisation de l'affichage lumineux de ton nom

Des circuits électriques différents peuvent réaliser les mêmes fonctions bien qu'ils soient très différents l'un de l'autre. Les LED en sont un exemple. Ainsi, tu peux utiliser des LED de différentes couleurs sans devoir changer ton circuit. Tu as cependant toujours besoin d'une résistance de limitation de courant et de liaisons au pôle positif de l'alimentation +5 V et à la masse GND.

Copyright © 2015 Eyrolles.

Les LED sont caractérisées par leur diamètre, tu peux en acheter de différentes tailles. Tu utilises probablement des LED de 5 mm, mais tu peux les acheter de toutes les tailles disponibles. Essaie par exemple les versions 3 mm et 10 mm. Les LED de 10 mm sont parfaites pour ce projet, mais tu peux utiliser n'importe quelle taille qui te paraîtra adéquate.

Dans ce projet, tu vas pouvoir allumer jusqu'à vingt-quatre LED. Tu décideras toi-même comment les utiliser pour embellir une enseigne lumineuse, par exemple une banderole de carnaval. Tu vas créer ta propre enseigne pour afficher ton nom avec des lampes (voir la figure 4-12).

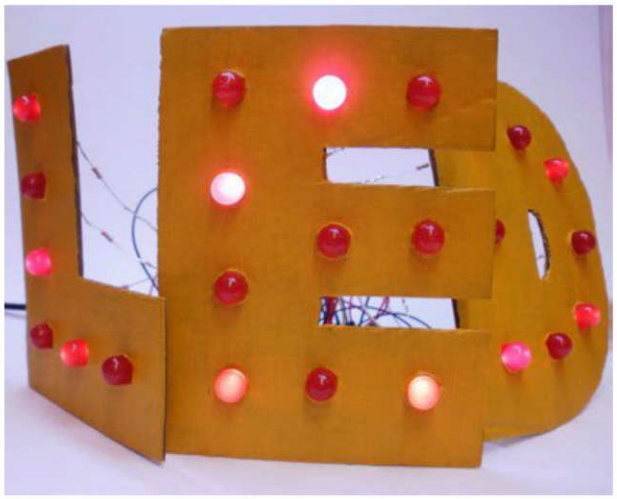


FIGURE 4-12 Ton nom (ou tout autre mot) en affichage lumineux.

Tu peux choisir l'affichage de ton choix. Ce peut être ton nom (ou simplement tes initiales, si ton nom est trop long) ou tout autre mot, comme « LED » !

Tu vas choisir les lettres que tu veux utiliser et les découper dans du carton. Tu décideras ensuite où placer les vingt-quatre LED et les ajouteras à tes lettres.

Tu peux regarder une vidéo (en anglais) montrant la réalisation d'un affichage lumineux sur le site www.editions-eyrolles.com/go/arduino.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

La liste suivante contient ce dont tu as besoin pour réaliser ton affichage :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage (tu en auras peut-être besoin de plus, si tu as beaucoup de lettres) ;
- 57 fils de pontage ;
- 24 LED ;
- 24 résistances de 220 Ω ;
- 3 registres à décalage 74HC595 (1 pour 8 LED) ;
- du papier cartonné (que tu peux aussi découper dans une boîte en carton) ;
- du fil de câblage ;
- de la soudure ;
- de la peinture ou du papier coloré pour les décors ;
- un fer à souder ;
- du ruban adhésif opaque ;
- des ciseaux ou un couteau de bricolage ;
- un crayon, un tournevis ou un poinçon.

Rappelle-toi que l'annexe A indique les endroits où tu peux acheter les composants présentés sur la figure 4-13.

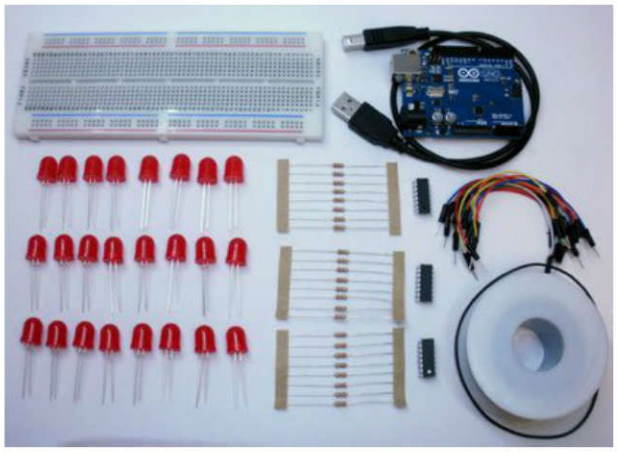


FIGURE 4-13 Les composants électroniques nécessaires à la réalisation de ton affichage lumineux.

Copyright © 2015 Eyrolles.

Comprendre le circuit

Le circuit de ce projet est très similaire au circuit précédent utilisant deux registres à décalage et seize LED. Tu peux placer jusqu'à trois registres à décalage et vingt-quatre LED. Tu as besoin d'un registre à décalage pour chaque groupe de 8 LED.

Il y a beaucoup de liaisons à effectuer. Utilise la figure 4-14 comme guide pour savoir ce que tu dois relier. Souviens-toi que seules les lignes qui se croisent avec un point sont des fils à relier entre eux.

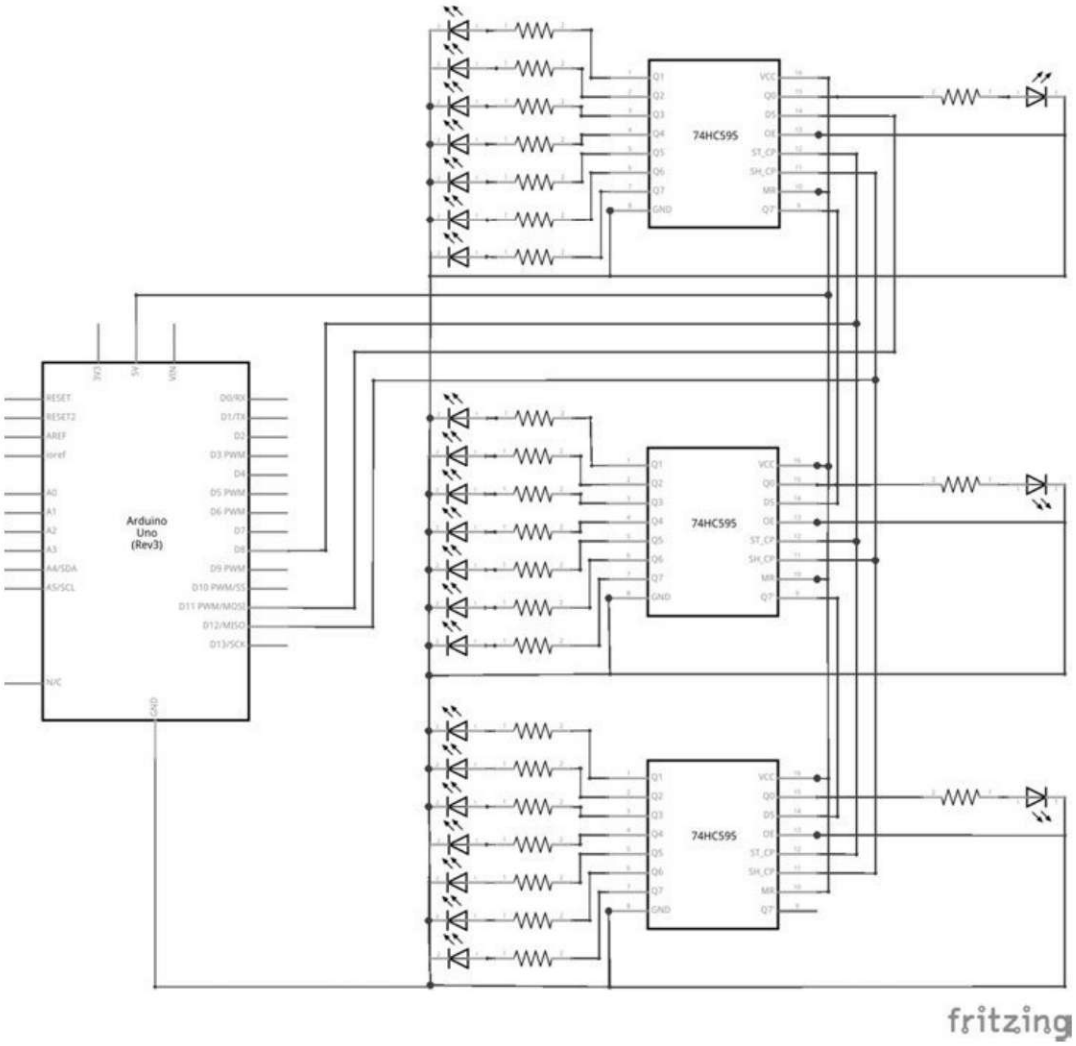


FIGURE 4-14 Le schéma du circuit à trois registres à décalage.

Copyright © 2015 Eyrolles.

Réalisation du prototype sur plaque de montage

Il faut toujours vérifier le fonctionnement correct d'un circuit avant de le réaliser définitivement en le soudant ! Construis ton prototype avec trois registres à décalage. Tu pourras avoir besoin de plusieurs plaques de montage. Suis les étapes de la section « Ajout d'autres registres à décalage » page 143 pour mettre en œuvre trois registres à décalage. La broche 14 d'un des registres à décalage sera connectée à la broche 11 de l'Arduino Uno. Pour les deux autres circuits, la broche 14 de l'un sera reliée à la broche 9 du suivant. Aide-toi de la figure 4-11 pour relier un troisième registre à décalage au deuxième, de la même façon que tu as relié le deuxième registre au premier. Connecte ensuite huit LED à chaque registre à décalage en suivant les étapes de la section « Ajout de LED » page 135.

Écriture du programme

Crée un nouveau sketch en cliquant sur le bouton Nouveau de l'IDE Arduino ou en utilisant le menu Fichier > Nouveau, puis entre le code suivant :

```
void setup() {
}
void loop() {
}
```

Tout en haut de ton sketch, avant l'instruction `setup()`, ajoute les variables suivantes.

La version originale :

```
// The pin connected to latch pin (ST_CP) of 74HC595
int latchPin = 8;
// The pin connected to clock pin (SH_CP) of 74HC595
int clockPin = 12;
// The pin connected to Data in (DS) of 74HC595
int dataPin = 11;
// The number of shift registers used
int numRegisters = 3;
// The first pattern to be displayed
int pattern1 = 85;
// The second pattern to be displayed
int pattern2 = 170;
```

La version francisée :

```
// La broche reliée à la broche LATCH (verrou) du 74HC595
int brocheVerrou = 8;
// La broche reliée à la broche CLOCK (horloge) du 74HC595
int brocheHorloge = 12;
```

Copyright © 2015 Eyrolles.

```
// La broche reliée à la broche LATCH (verrou) du 74HC595
int brocheDonnees = 11;
// Le nombre de registres à décalage utilisés
int nombreRegistres = 3;
// La première combinaison à afficher
int combinaison1 = 85;
// La deuxième combinaison à afficher
int combinaison2 = 170;
```

À l'intérieur de la fonction `setup()`, définis le mode de fonctionnement des broches puis appelle la fonction `setLEDs()` ou `commandeLED()` que tu vas ensuite écrire. Ajoute le code suivant :

La version originale :

```
// Set pins to output because they are addressed in the main
// loop
pinMode(latchPin, OUTPUT);
pinMode(dataPin, OUTPUT);
pinMode(clockPin, OUTPUT);
// Start with all LEDs off
setLEDs(0);
```

La version francisée :

```
// Déclaration en mode sortie des broches utilisées dans la
// boucle principale
pinMode(brocheVerrou, OUTPUT);
pinMode(brocheDonnees, OUTPUT);
pinMode(brocheHorloge, OUTPUT);
// Démarrage avec toutes les LED éteintes
commandeLED(0);
```

Dans la fonction `loop()`, entre le code suivant pour envoyer une combinaison faisant clignoter les LED.

La version originale :

```
// Turn on the LEDs in the pattern 01010101
setLEDs(pattern1);
// Wait 1 sec
delay(1000);
// Turn on the LEDs in the pattern 10101010
setLEDs(pattern2);
// Wait 1 sec
delay(1000);
```

Copyright © 2015 Eyrolles.

La version francisée :

```
// Activer les LED selon la combinaison 01010101
commandeLED(combinaison1);
// Attendre 1 seconde
delay(1000);
// Activer les LED selon la combinaison 10101010
commandeLED(combinaison2);
// Attendre 1 seconde
delay(1000);
```

La seule fonction manquante est `setLEDs()` ou `commandeLED()`. C'est une nouvelle fonction que tu dois écrire, car elle n'existe pas dans l'IDE Arduino. Entre le code suivant après le dernier crochet `}` de la fonction `loop()`. Tu trouveras plus d'informations sur ce code dans l'encadré « Exploration du code » page 154.

La version originale :

```
// Send pattern to shift register for
// which LEDs to turn on and off
void setLEDs(int lightPattern) {
  // Turn off the output so the pins don't light up
  // while you're shifting bits
  digitalWrite(latchPin, LOW);

  int i;
  for(i=0; i<numRegisters; i++) {
    // Send out the pattern once for each shift register
    shiftOut(dataPin, clockPin, LSBFIRST, lightPattern);
  }
  // Turn on the output so the LEDs can light up
  digitalWrite(latchPin, HIGH);
  delay(300);
}
```

La version francisée :

```
// Envoie les combinaisons au registre à décalage
// pour allumer ou éteindre les LED
void commandeLED(int combinaisonLED) {
  // Force le verrouillage des sorties afin que les
  // LED restent éteintes pendant le décalage des bits
  digitalWrite(brocheVerrou, LOW);

  int i;
  for(i=0; i<nombreRegistres; i++) {
    // Envoie la combinaison à chaque registre à décalage
```

Copyright © 2015 Eyrolles.

```
    shiftOut(brocheDonnees, brocheHorloge, LSBFIRST,
combinaisonLED);
  }
  // Déverrouille les sorties afin d'activer les LED :
  digitalWrite(brocheVerrou, HIGH);
  delay(300);
}
```

Voici le sketch complet, que tu peux aussi télécharger en anglais ou en français depuis le site d'accompagnement www.editions-eyrolles.com/go/arduino.

La version originale :

```
// The pin connected to latch pin (ST_CP) of 74HC595
int latchPin = 8;
// The pin connected to clock pin (SH_CP) of 74HC595
int clockPin = 12;
// The pin connected to Data in (DS) of 74HC595
int dataPin = 11;
// The number of shift registers used
int numRegisters = 3;
// The first pattern to be displayed
int pattern1 = 85;
// The second pattern to be displayed
int pattern2 = 170;

void setup() {
  // set pins to output because they are addressed in the
  // main loop
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);

  // start with all LEDs off
  setLEDs(0);
}

void loop() {
  // turn on LEDs in the pattern 01010101
  setLEDs(pattern1);
  // wait 1 sec
  delay(1000);
  // turn on LEDs in the pattern 10101010
  setLEDs(pattern2);
  // wait 1 sec
  delay(1000);
}
```

Copyright © 2015 Eyrolles.

```
// Send pattern to shift register for
// which LEDs to turn on and off
void setLEDs(int lightPattern) {
  // turn off the output so the pins don't light up
  // while you're shifting bits:
  digitalWrite(latchPin, LOW);
  int i;
  for(i=0; i<numRegisters; i++) {
    // send out the pattern once for each shift register
    shiftOut(dataPin, clockPin, LSBFIRST, lightPattern);
  }
  // turn on the output so the LEDs can light up:
  digitalWrite(latchPin, HIGH);
  delay(300);
}
```

La version francisée :

```
// La broche reliée à la broche LATCH (verrou) du 74HC595
int brocheVerrou = 8;
// La broche reliée à la broche CLOCK (horloge) du 74HC595
int brocheHorloge = 12;
// La broche reliée à la broche LATCH (verrou) du 74HC595
int brocheDonnees = 11;
// Le nombre de registres à décalage utilisés
int nombreRegistres = 3;
// La première combinaison à afficher
int combinaison1 = 85;
// La deuxième combinaison à afficher
int combinaison2 = 170;

void setup() {
  // Déclaration en mode sortie des broches utilisées dans
  // la boucle principale
  pinMode(brocheVerrou, OUTPUT);
  pinMode(brocheDonnees, OUTPUT);
  pinMode(brocheHorloge, OUTPUT);

  // Démarrage avec toutes les LED éteintes
  commandeLED(0);
}
void loop() {
  // Activer les LED selon la combinaison 01010101
  commandeLED(combinaison1);
  // Attendre 1 seconde
  delay(1000);
  // Activer les LED selon la combinaison 10101010
```

Copyright © 2015 Eyrolles.

```
  commandeLED(combinaison2);
  // Attendre 1 seconde
  delay(1000);
}

// Envoie les combinaisons au registre à décalage
// pour allumer ou éteindre les LED
void commandeLED(int combinaisonLED) {
  // Force le verouillage des sorties afin que les
  // LED restent éteintes pendant le décalage des bits :
  digitalWrite(brocheVerrou, LOW);
  int i;
  for(i=0; i<nombreRegistres; i++) {
    // Envoie la combinaison à chaque registre à décalage
    shiftOut(brocheDonnees, brocheHorloge, LSBFIRST,
  combinaisonLED);
  }
  // Déverrouille les sorties afin d'activer les LED :
  digitalWrite(brocheVerrou, HIGH);
  delay(300);
}
```

EXPLORATION DU CODE

Les déclarations des variables au début du programme et le contenu de la fonction `setup()` doivent te paraître familiers, ils sont identiques à ce que tu as précédemment utilisé dans cette aventure. Une nouvelle variable est apparue : `numRegisters` ou `nombreRegistres`. Elle a pour valeur `3`, mais si tu souhaites utiliser moins de registres à décalage, tu peux en changer la valeur selon la quantité utilisée.

Il y a aussi une nouvelle fonction `setLEDs()` ou `commandeLED(`, qui est définie après `loop()`. Elle reçoit un argument qui est la combinaison indiquant les LED qui doivent être allumées (`pattern1` ou `pattern2` dans la version anglaise, `combinaison1` ou `combinaison2` dans la version francisée).

La combinaison est exprimée par un nombre binaire. Pour `pattern1` ou `combinaison1`, c'est la valeur décimale 85 qui correspond à 01010101 en codage binaire, elle permet d'allumer les LED paires de chaque ensemble de huit LED. L'autre combinaison `pattern2` ou `combinaison2` vaut 170 en décimal ou 1010101010 en binaire. C'est l'inverse de la première combinaison : elle permet l'allumage des LED qui étaient éteintes avec la première combinaison. Ainsi, quand on alterne les combinaisons `pattern1` ou `combinaison1` et `pattern2` ou `combinaison2`, il en résulte un effet de clignotement.

Copyright © 2015 Eyrolles.

Tu souhaites faire clignoter ton affichage de façon différente ? Cela n'est pas trop difficile : il te suffit de créer tes propres combinaisons. Écris chaque combinaison en utilisant une série de huit bits 0 ou 1 et calcule ensuite sa valeur décimale. Tu peux aussi utiliser un convertisseur binaire décimal en ligne, comme le site www.fr.calculworld.com/calculs-mathematiques/calculatrice-hexadecimal.

Fabrication de tes lettres lumineuses

La première et importante décision que tu dois prendre est le choix des lettres que tu vas créer. Rappelle-toi que tu n'as que vingt-quatre LED pour décorer ton affichage. Tu dois donc décider les lettres que tu souhaites réaliser et où les LED y seront placées.

1. Trace tes lettres sur du papier cartonné : du carton récupéré sur une ancienne boîte, par exemple, conviendra parfaitement. Utilise une paire de ciseaux ou un couteau de bricolage pour découper.
2. Si tu veux décorer tes lettres avec de la peinture ou avec du papier, fais-le maintenant.
3. Quand la peinture et la colle sont sèches, utilise un outil pour percer des trous dans le carton aux endroits où tu souhaites placer chaque LED : un crayon ou un tournevis conviendront parfaitement pour cela. Assure-toi que les trous soient assez gros pour maintenir les LED en place. À cette étape, tu as déterminé l'emplacement des LED. Tu vas insérer les LED dans leurs trous après les avoir soudées à leurs fils. Tu dois maintenant avoir des lettres similaires à celles de la figure 4-15.

FIGURE 4-15 Des lettres en papier cartonné avec des trous pour les LED.

Soudage des fils

Suis les étapes suivantes pour souder ton circuit.

1. Soude une résistance au fil le plus long de chaque LED. Torsade ensemble les fils de chaque résistance et de chaque LED pour qu'ils soient fermement fixés l'un à l'autre, puis soude-les.

Copyright © 2015 Eyrolles.

Un adulte doit toujours être à proximité quand tu soudes ton circuit. Consulte les recommandations de l'aventure 2 pour effectuer les opération de soudure en toute sécurité.

2. Place les LED dans les trous des lettres en carton. Plie le fil le plus court de chaque LED vers la LED suivante (guide-toi en regardant la figure 4-16). Si le fil court de chaque LED ne peut pas rejoindre le fil court de la suivante, coupe un petit morceau de fil pour les joindre. Soude le fil ajouté aux deux LED : le fil court de la première LED au fil court de la LED suivante. Répète cela pour toutes les LED. Tu dois avoir un fil court d'une seule LED laissé libre sur chaque lettre.
3. Utilise une plaque de montage pour relier tes registres à décalage à tes LED et à ta carte Arduino Uno. Décide de l'endroit où tu vas placer ta plaque de montage et ta carte Arduino. Elles peuvent être collées au dos des lettres à l'aide de ruban adhésif ou rester à proximité sur ta table de travail.
4. Découpe 24 morceaux de fil pour relier chacune des résistances soudées à tes LED à la plaque de montage. Tu pourrais souhaiter cacher les fils en les plaquant au dos des lettres avec du ruban adhésif. Dans ce cas, fais attention de bien découper des fils suffisamment longs.
5. Dénude environ 12 mm d'isolant à chaque extrémité des fils que tu viens de découper. Soude chaque fil à la résistance de chaque LED.
6. Découpe un morceau de fil pour relier le fil court resté libre de la dernière LED de chaque lettre à la plaque de montage. Tu dois en découper un pour chaque lettre.
7. Dénude environ 12 mm d'isolant à chaque extrémité des fils que tu viens de découper. Soude chacun de ces fils au fil court resté libre de la dernière LED de chaque lettre.

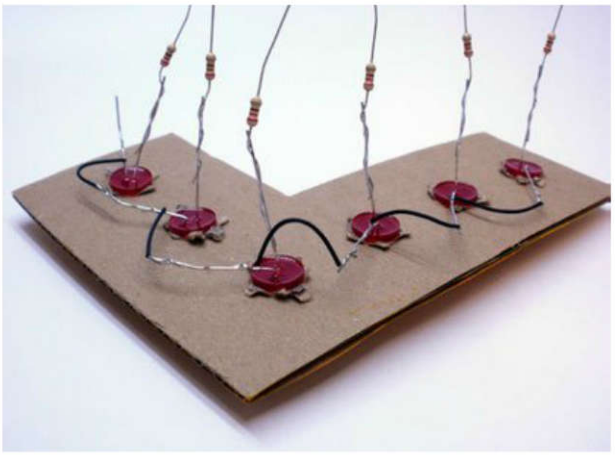


FIGURE 4-16 Une vue des LED et des résistances soudées.

Copyright © 2015 Eyrolles.

Insertion du circuit électronique

Quand tu as fini de souder toutes tes LED, insère-les dans leurs trous sur chaque lettre. Tu auras peut-être besoin d'ajouter un peu de colle si elles ne tiennent pas fermement en place. Tu peux maintenant construire le circuit de tes registres à décalage sur ta plaque de montage comme illustré sur la figure 4-17.

1. Suis les mêmes étapes que lorsque tu as réalisé le prototype de ton circuit. Connecte les liaisons d'alimentation +5 V, de masse GND, `CLOCK` (l'horloge), `LATCH` (le verrouillage) et `DATA` (les données) comme tu l'as fait précédemment.
2. Les résistances sont soudées aux LED, elles ne doivent donc pas être placées sur la plaque de montage. Au lieu de cela, relie les fils que tu as soudés aux résistances aux broches des registres à décalage. Regarde la figure 4-16 pour t'aider.
3. Relie les fils provenant des fils les plus courts des LED à la masse GND de la plaque de montage.
4. Connecte maintenant ton Arduino Uno à ton ordinateur ou à une source d'alimentation.

Applique la tension d'alimentation à ta carte Arduino Uno et tu constateras qu'immédiatement ton nom s'illumine !

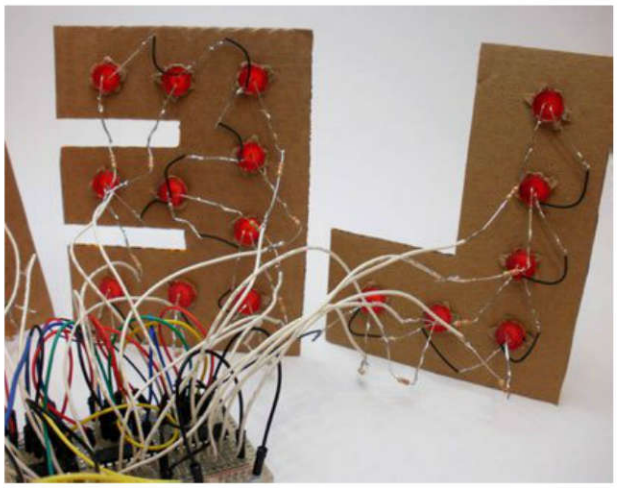


FIGURE 4-17 Vue du dos des lettres.

Plus d'aventures avec les registres à décalage

Maintenant que tu as réalisé un magnifique affichage, tu peux essayer différentes animations lumineuses. Essaie d'écrire tes propres fonctions avec les combinaisons que tu vas créer.

Copyright © 2015 Eyrolles.

Tu pourrais aussi prendre connaissance de ces différents articles (en anglais) en ligne :

- `http://arduino.cc/en/tutorial/ShiftOut`
- `https://learn.adafruit.com/adafruit-arduino-lesson-4-eight-leds/the-74hc595-shift-register`

Dans le projet que tu viens de réaliser, tu envoies un nombre décimal à la fonction `shiftOut()` qui le convertit en valeur binaire pour indiquer à chaque broche de sortie si elle doit être au niveau haut (`HIGH`) ou au niveau bas (`LOW`). Tu peux en apprendre davantage sur les nombres binaires en allant sur le site en français ou `https://fr.wikipedia.org/wiki/Binary_coded_decimal`.

Rappel des commandes Arduino de cette aventure

Commande	Description
`void`	Indique à l'ordinateur qu'aucune donnée ne sera retournée par une fonction quand son exécution se termine. Voir également `http://arduino.cc/en/Reference/Void`.
`shiftOut()`	Envoie une série de niveaux haut (`HIGH`) et bas (`LOW`) en synchronisation avec le signal `CLOCK` (l'horloge). Voir aussi `http://arduino.cc/en/Reference/shiftOut`.

Étape franchie : Tu es maintenant un brillant ingénieur en affichage lumineux !

Dans la prochaine aventure

Tu apprendras à ajouter un haut-parleur et à jouer de la musique en transformant ton Arduino Uno en synthétiseur électronique !

Copyright © 2015 Eyrolles.

Aventure 5

Diffuse des sons

IL EXISTE PLUSIEURS façons de contrôler des objets du monde réel avec un programme. Tu as déjà contrôlé un mouvement avec des moteurs et une lumière avec des LED. Dans cette aventure, tu vas créer des effets sonores !

Dans les aventures 3 et 4, tu as appris à rendre ton code plus efficace lorsque tu dois exécuter des instructions plus d'une fois. La boucle `for` est un outil génial pour répéter des actions un certain nombre de fois. En la combinant avec une liste particulière du code appelée « tableau », tu obtiens un outil de programmation très puissant.

Tu vas maintenant apprendre à maîtriser la puissance des tableaux. Puis tu vas découvrir comment ton Arduino Uno peut interpréter un air musical pour toi. Tu vas ensuite mettre tes connaissances en pratique en réalisant un carillon à vent amélioré qui diffusera des sons acoustiques et électroniques.

Ce dont tu as besoin

Tu as besoin de peu d'éléments pour la première partie de cette aventure (les composants électroniques nécessaires sont présentés sur la figure 5-1) :

- un ordinateur ;
- une carte Arduino Uno ;

Copyright © 2015 Eyrolles.

- un câble USB ;
- une plaque de montage ;
- 7 fils de pontage ;
- 6 LED ;
- 6 résistances de 220 Ω ;
- un diffuseur piézoélectrique.

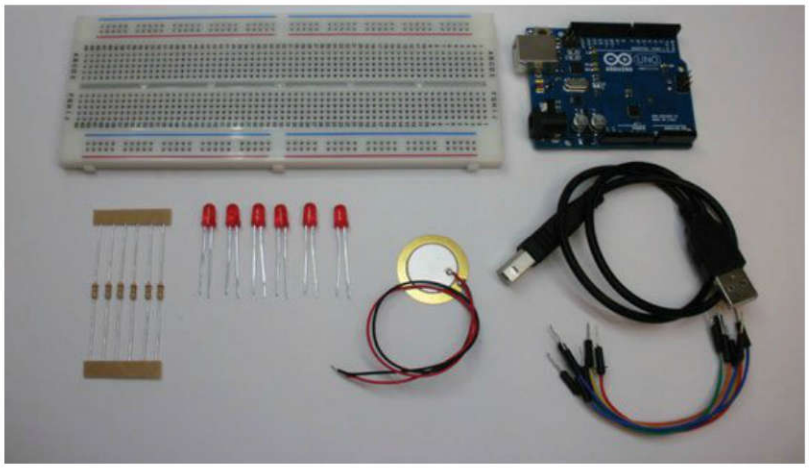


FIGURE 5-1 Les composants électroniques nécessaires pour la première partie de cette aventure.

Création d'une liste

Les variables sont très utiles pour mémoriser des informations, comme les numéros des broches de ta carte Arduino Uno. En créant une variable pour mémoriser le numéro de la broche à laquelle une LED est reliée, tu rends ton code plus facile à lire par la suite. Quand tu crées une variable comme ceci :

```
int brocheLed = 12 ;
```

tu peux ensuite écrire :

```
digitalWrite(brocheLed, HIGH) ;
```

au lieu de :

```
digitalWrite(12, HIGH) ;
```

L'utilisation de variables rend ton code plus facile à lire. Tu pourrais ne plus te rappeler de ce qui est relié à la broche 12, mais la variable `brocheLed` rend plus évident ce que fait le code.

Copyright © 2015 Eyrolles.

Mais comment conserver la trace de plus d'une LED ? Eh bien, tu pourrais créer une variable pour chaque broche de la façon suivante :

```
int brocheLed1 = 3 ;
int brocheLed2 = 4 ;
int brocheLed3 = 5 ;
int brocheLed4 = 6 ;
int brocheLed5 = 3 ;
int brocheLed6 = 8 ;
```

Mais cela ne semble pas très efficace ! Pourquoi devrais-tu écrire la même chose plusieurs fois, alors que tu sais qu'un ordinateur sait très bien le faire ? Il doit y avoir une meilleure idée, pas vrai ?

Oui, en effet ! Les ordinateurs, dont Arduino, peuvent enregistrer des informations sous la forme de listes appelées **tableaux.** Au lieu de créer une variable pour chaque information, tu en crées une seule qui est un tableau. Chaque fois que tu veux te référer à l'une des valeurs, tu dois simplement indiquer sa position dans le tableau. La figure 5-2 montre l'exemple de deux tableaux.

La seule astuce dont il faut se souvenir est que le premier élément d'un tableau n'est pas le numéro 1 mais le numéro 0. Par exemple, si tu veux utiliser le premier élément d'une liste appelée `brochesLED`, tu devras écrire :

```
brochesLED[0]
```

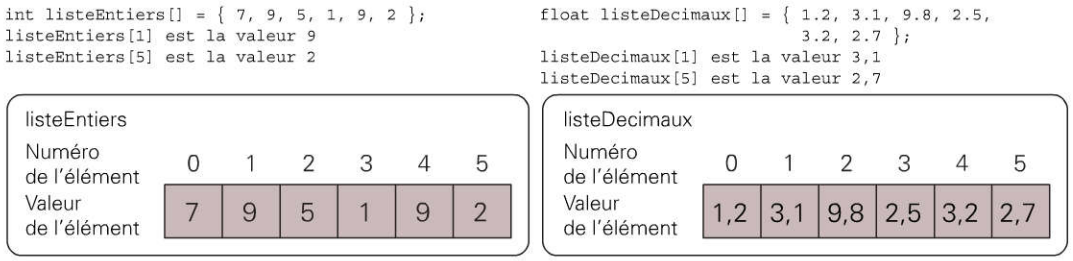


FIGURE 5-2 Deux exemples de tableaux de valeurs.

Un **tableau** dans un programme est une liste d'éléments de même type. Par exemple, un tableau peut contenir une liste de valeurs entières de type `int`.

Si tu veux allumer la LED reliée à la broche dont le numéro est mémorisé dans la troisième position du tableau, tu dois écrire :

```
digitalWrite(brochesLED[2], HIGH) ;
```

Copyright © 2015 Eyrolles.

Dévoile tes intentions

Comment peux-tu créer un nouveau tableau ? Tu ne dois pas simplement créer une variable contenant une seule valeur entière `int` ou flottante **float**. La méthode comprend davantage d'étapes.

Un **nombre flottant**, de type **float**, est utilisé pour exprimer les nombres qui ne sont pas des entiers mais incluent des chiffres décimaux, par exemple 1,3 ou -54,089. Mais attention, en programmation, un point doit être utilisé à la place de la virgule.

La première étape consiste à déclarer une nouvelle variable qui va contenir le tableau. **Déclarer** est un mot un peu curieux pour désigner quelque chose que tu as déjà fait précédemment dans tes sketches. Cela signifie créer une nouvelle variable en lui donnant un nom et un type de données. La ligne de code suivante déclare une nouvelle variable appelée `brocheLed` contenant un nombre entier `int` :

```
int brocheLed ;
```

Déclarer une variable consiste à créer une nouvelle variable en lui donnant un nom et un type de données tel que `int`. La variable ne contiendra une valeur qu'après que celle-ci lui sera affectée.

Si tu connais déjà la valeur que tu veux mémoriser dans la variable que tu crées, tu peux l'**initialiser** en même temps que tu la déclares. Cela signifie que tu lui donnes une valeur de départ :

```
int brocheLed = 13 ;
```

Initialiser (instancier ou *instanciate* en anglais) une variable consiste à lui donner une valeur pour la première fois. L'initialisation de la variable peut se produire en même temps que sa déclaration ou plus tard dans le programme. La déclaration doit toujours être réalisée en premier.

Tu n'es pas obligé de déclarer et d'initialiser la variable au même moment : tu peux faire tout cela par la suite dans ton programme.

Copyright © 2015 Eyrolles.

```
int brocheLed ; // déclaration (instanciation)
// instructions quelconques suivant cette déclaration
brocheLed = 13 ;
```

Fais bien attention : si tu décides de ne pas initialiser une variable au moment de sa déclaration, tu ne pourras pas l'utiliser avant qu'elle ne possède une valeur. Sinon, ton programme Arduino pourrait ne pas fonctionner comme tu le souhaites.

Pour déclarer un nouveau tableau de six valeurs entières appelé `listeLED`, tu dois écrire le code suivant :

```
int listeLED[6] ;
```

L'instruction précédente est légèrement différente de ce que tu devais écrire pour créer une nouvelle variable contenant un nombre entier. Après le nom de la variable `listeLED`, se trouve `[6]`. Les crochets `[ ]` signifient qu'au lieu d'être une simple variable, il s'agit d'un tableau de valeurs entières `int`. Le chiffre `6` indique le nombre d'entiers que le tableau va contenir.

Maintenant que tu as créé un nouveau tableau, tu peux l'initialiser en le remplissant de nombres dans la suite dans ton code :

```
listeLED[0] = 3 ;
listeLED[1] = 4 ;
listeLED[2] = 5 ;
listeLED[3] = 6 ;
listeLED[4] = 7 ;
listeLED[5] = 8 ;
```

Déclarer un tableau sans l'initialiser en même temps est pratique quand tu ne connais pas les valeurs qu'il doit contenir au moment où tu le crées. Mais si tu les connais déjà, tu peux l'initialiser au moment de sa déclaration comme indiqué ci-dessous :

```
int listeLED [] = {3, 4, 5, 6, 7, 8} ;
```

Les valeurs mémorisées dans le tableau sont listées entre les accolades `{ }`. Il n'est alors plus nécessaire d'indiquer une valeur entre les crochets `[]`, car la taille du tableau est fixée par le nombre de valeurs d'initialisation à l'intérieur des accolades `{ }`.

Copyright © 2015 Eyrolles.

Utilise une boucle et un tableau

Il est facile d'exécuter une suite d'actions sur chaque valeur d'un tableau sans avoir à répéter les mêmes instructions à plusieurs reprises. Tu peux en effet utiliser une boucle `for` pour cela. Comme tu l'as vu dans les aventures précédentes, une boucle `for` contient trois informations lui permettant de déterminer le nombre de fois où elle s'exécute. Une nouvelle variable, souvent nommée `i`, est créée et utilisée pour compter au fur et à mesure le nombre d'exécutions de la boucle. Dans le code suivant, la boucle va s'exécuter six fois :

```
int i ;
for (i=0 ; i<6 ; i++) { }
```

`i++` indique que la variable `i` s'incrémente de 1 à chaque fois que l'ordinateur parcourt la boucle. Au lieu d'accéder à une seule valeur du tableau comme `ledList[3]`, la variable `i` peut être utilisée pour accéder à la valeur suivante de la liste, chaque fois que le programme parcourt la boucle. Voici un exemple :

```
int i ;
for (i=0 ; i<6 ; i++) {
  Serial.println( listeLED[i] ) ; // afficher la prochaine
  // valeur de la liste
}
```

Tu peux également écrire une boucle `for` qui modifie chaque valeur du tableau :

```
int i ;
for (i=0 ; i<6 ; i++) {
  listeLED[i] = i+2 // écrire la valeur i+2 dans chaque
  // position du tableau
}
```

Mets cela en pratique

Assez parlé de ce qui se passe dans le code, il est temps d'allumer quelques LED et de voir ce qui se passe dans le programme !

Le circuit est celui que tu as déjà vu plusieurs fois au cours des aventures précédentes : une LED avec sa résistance de limitation de courant. Cette fois-ci, tu vas en utiliser six sur les broches numériques 3 à 8, comme indiqué sur la figure 5-3.

1. Commence en utilisant un fil de pontage pour relier la masse GND de la carte Arduino Uno à l'une des longues rangées en bas de ta plaque de montage. Si ta plaque comporte un marquage bleu ou noir ou le signe -, relie la masse à cette rangée.

Copyright © 2015 Eyrolles.

EXPLORATION DU CODE

La mini-instruction `i++` est un raccourci de `i=i+1`, mais ce n'est pas la seule mini-instruction pratique. Si `i++` incrémente `i` en lui ajoutant `1` puis en mémorisant cette nouvelle valeur dans `i`, tu dois deviner ce que `i--` fait. Oui, c'est ça, elle soustrait `1` de `i` et mémorise le résultat dans `i`.

Une autre façon d'incrémenter `i` de `1` est la mini instruction `i+=1`. Tu peux utiliser ces conventions pour augmenter ou diminuer une variable. Ainsi, pour augmenter `i` de `3`, tu écrirais `i+=3`, et pour la diminuer de `7`, tu écrirais `i-=7`.

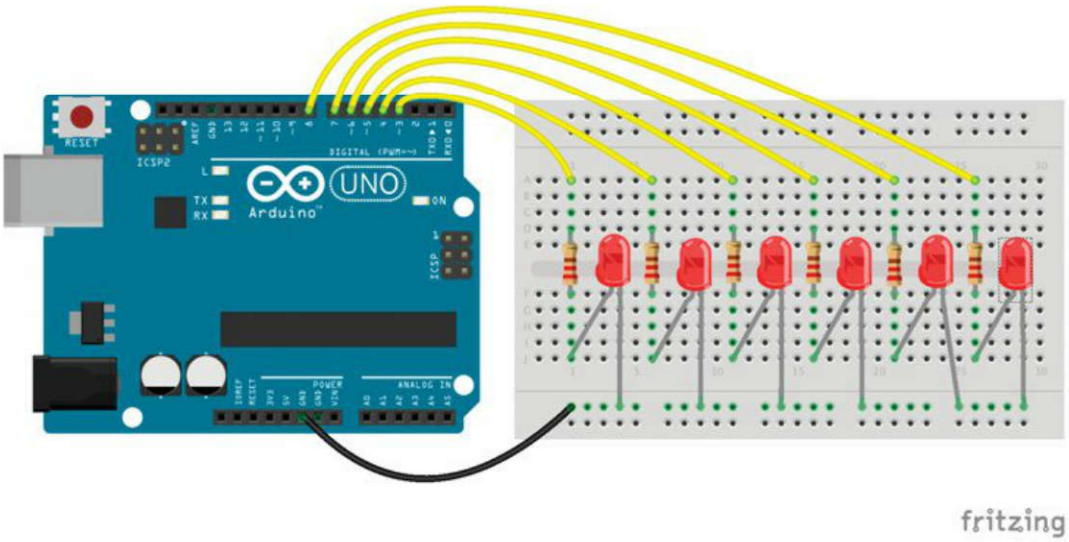


FIGURE 5-3 Le circuit pour un ensemble de LED.

2. Place tes six LED sur ta plaque de montage, en insérant leur fil le plus court dans la longue rangée reliée à la masse GND. Répartis-les uniformément sur la plaque.
3. Insère le fil le plus long de chaque LED dans la rangée courte la plus proche. La rangée que tu utilises n'a aucune importance : ce qui compte, c'est de faire en sorte que chaque LED se trouve dans une rangée différente.
4. Relie un fil d'une résistance dans chaque rangée où une LED est placée. Plie les fils de la résistance de façon à ce qu'elle se trouve au-dessus de la séparation du milieu de la plaque, insère alors l'autre fil de chaque résistance dans la rangée, de l'autre côté de la séparation.
5. Relie chaque résistance à une sortie de la carte Arduino Uno. Utilise pour cela six fils de pontage pour connecter les résistances aux sorties 3 à 8 de la carte Arduino Uno.

Copyright © 2015 Eyrolles.

Le programme suivant te permet d'utiliser d'un tableau de valeurs. Ici, chacune des positions du tableau est le numéro d'une broche qui contrôle une LED. Tu peux donc faire clignoter le groupe de LED de la même façon que tu en ferais clignoter une seule. La fonction `pinMode()` est utilisée pour chacune des broches : chacune reçoit la valeur `HIGH` (niveau HAUT) puis `LOW` (niveau BAS) :

La version originale en anglais :

```
// Pins for the leds
int ledList[] = {
  3, 4, 5, 6, 7, 8};
// The number of pins
int numPins = 6;

void setup() {
  int i;
  for(i=0; i<numPins; i++){
    // set pins to OUTPUT
    pinMode(ledList[i], OUTPUT);
  }
}

void loop() {
  // blink the LEDs one by one
  int i;
  for(i=0; i<numPins; i++) {
    // turn on the led
    digitalWrite(ledList[i], HIGH);
    delay(500);
    // turn off the led
    digitalWrite(ledList[i], LOW);
    delay(500);
  }
}
```

La version francisée :

```
// Les broches des LED
int listeLED[] = {3, 4, 5, 6, 7, 8};

// Le nombre de broches
int nombreBroches = 6;

void setup() {
  int i;
  for(i=0; i<nombreBroches; i++){
    // initialiser les broches en mode sortie OUTPUT
```

Copyright © 2015 Eyrolles.

```
    pinMode(listeLED[i], OUTPUT);
  }
}

void loop() {
  // faire clignoter les LED une par une
  int i;
  for(i=0; i<nombreBroches; i++) {
    // allumer la LED
    digitalWrite(listeLED[i], HIGH);
    delay(500);
    // éteindre la LED
    digitalWrite(listeLED[i], LOW);
    delay(500);
  }
}
```

Que se passe-t-il si tu changes la boucle `for` dans la fonction `loop()` de cette façon :

```
for (i=nombreBroches-1 ; i>0 ; i--) ?
```

EXPLORATION DU CODE

Tes six LED clignotent maintenant une par une ; mais comment le programme provoque-t-il cela ? Commençons par le début du sketch. Tu vois les variables qui sont utilisées ? La première est un tableau `ledList` (ou `listeLED`), qui contient les numéros de toutes les broches des LED. La seconde est un entier `int numPins` (ou `nombreBroches`) qui contient le nombre total de broches de sortie vers les LED.

```
// Les broches des LED
int listeLED[] = {3, 4, 5, 6, 7, 8};

// Le nombre de broches
int nombreBroches = 6;
```

En parcourant le programme, tu parviens à la première boucle `for` du sketch, située dans la fonction `setup()`. Cette boucle est utilisée pour initialiser les broches numériques de la carte Arduino Uno en mode sortie (`OUTPUT`). La variable `i` sert à accéder à chaque valeur du tableau :

Copyright © 2015 Eyrolles.

```
int i;
for(i=0; i<nombreBroches; i++){
  // initialiser les broches en mode sortie OUTPUT
  pinMode(listeLED[i], OUTPUT);
}
```

La deuxième boucle `for` de ce sketch est située dans la fonction `loop()`. La variable `i` est utilisée à nouveau pour parcourir le tableau. Cette fois, au lieu de définir le mode de fonctionnement des broches numériques, chacune d'elles est mise au niveau haut (`HIGH`). L'Arduino Uno attend ensuite 500 ms, lorsque `delay(500)` est appelée, puis la broche est mise au niveau bas (`LOW`). La Uno fait ensuite une nouvelle pause de 500 ms, avant de passer au numéro de la broche suivante du tableau.

```
// faire clignoter les LED une par une
int i;
for(i=0; i<nombreBroches; i++) {
  // allumer la LED
  digitalWrite(listeLED[i], HIGH);
  delay(500);
  // éteindre la LED
  digitalWrite(listeLED[i], LOW);
  delay(500);
}
```

Cette boucle se situe à l'intérieur de la fonction `loop()`. Elle s'exécutera donc indéfiniment jusqu'à ce que la carte Arduino Uno ne soit plus alimentée électriquement.

Production de bruit

Ce n'est pas inhabituel pour un ordinateur de produire des sons. Tu écoutes sûrement de la musique enregistrée sur un petit appareil reproduisant de la musique, ou sur ton téléphone (qui sont des sortes d'ordinateurs). Mais tu n'as probablement pas imaginé comment un ordinateur peut créer physiquement le son que tu entends...

Les sons ne sont que des vibrations de l'air que tes oreilles détectent. La vitesse selon laquelle les vibrations se produisent détermine si le son est grave ou aigu. Les vibrations sont mesurées en hertz (Hz) : un hertz équivaut à une vibration par seconde. Les êtres humains peuvent normalement entendre les sons situés entre 20 Hz et 20 000 Hz. Mais plus on vieillit et moins on entend les sons aigus des fréquences hautes, contrairement aux personnes jeunes. La figure 5-4 illustre la façon dont un son est produit.

Copyright © 2015 Eyrolles.

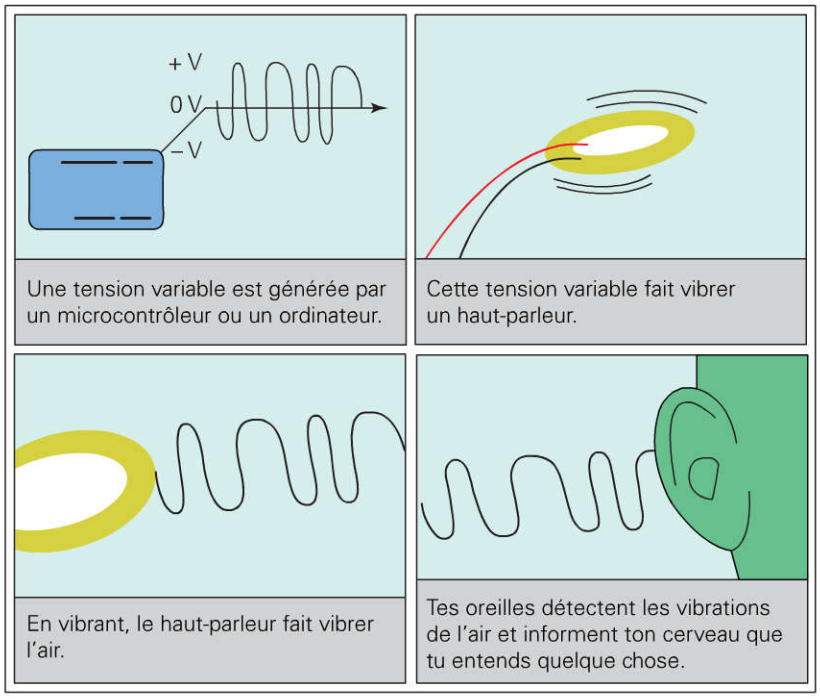


FIGURE 5-4 Reproduction d'un son.

Bien, mais comment un ordinateur fait-il vibrer l'air ? Un ordinateur ou une carte Arduino Uno peuvent générer une tension qui oscille entre des valeurs positives et négatives. Un haut-parleur utilise cette tension variable et la transforme en vibrations. Certains types de haut-parleurs utilisent un **élément piézoélectrique**. Cela ressemble généralement à un disque doré duquel sortent deux fils, le tout placé à l'intérieur d'un boîtier en plastique.

> Un **élément piézoélectrique** est un cristal qui se dilate ou se rétracte quand un courant électrique le traverse. Il peut aussi générer une tension électrique lorsqu'il est compressé ou plié.
>
>
>

Maintenant que tu sais à peu près comment un ordinateur produit des sons, tu peux faire en sorte que ton Arduino Uno en produise quelques-uns. Tu sais que ta carte génère des tensions de 0 et 5 V en utilisant la fonction `digitalWrite()`, qui permet de produire un niveau bas (`LOW`, 0 V) ou haut (`HIGH`, +5 V). Pour qu'une carte Arduino Uno puisse générer une tension comprise entre 0 et 5 V, on utilise une astuce. Par chance, les bibliothèques de code Arduino contiennent des fonctions qui le font et permettent de générer des sons sans que tu aies besoin d'en connaître tous les détails. Bien entendu, si tu veux en savoir plus sur la façon dont les sons sont reproduits, tu peux consulter la documentation Arduino (en anglais) à l'adresse `http://arduino.cc/en/Reference/Tone`.

Copyright © 2015 Eyrolles.

Câblage de ton circuit

Un reproducteur piézoélectrique possède deux fils. Le fil noir doit être relié à la masse GND de ta carte Arduino Uno et le fil rouge à la broche qui va générer le signal sonore. Relie le fil noir à la masse GND et le fil rouge à la broche 8, comme sur la figure 5-5.

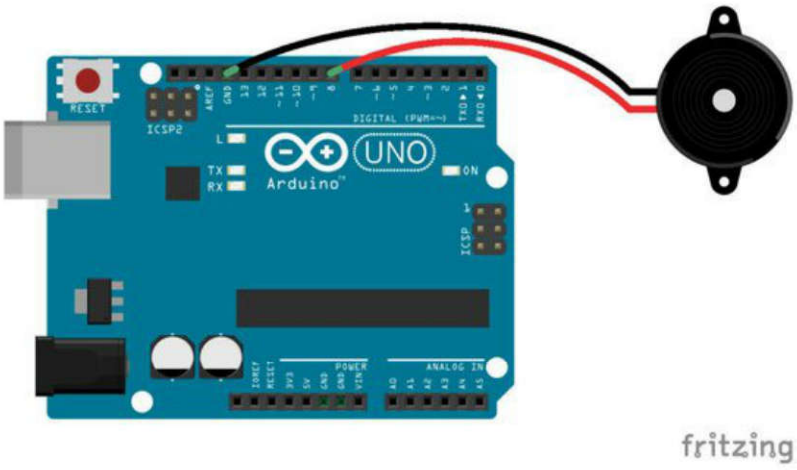


FIGURE 5-5 Le circuit utilisant un haut-parleur piézoélectrique.

Écriture du programme

La fonction principale utilisée pour générer un son est `tone()`. Elle peut être utilisée de deux manières différentes. La première consiste à lui fournir trois arguments : le numéro de la broche à partir de laquelle le son sera reproduit, la fréquence du son et la durée de reproduction de ce son. En voici un exemple :

```
// Reproduire un son de 750 Hz
// pendant 1 seconde par la broche 8
tone (8, 750, 1000) ;
delay (1000) ;
```

Tu peux remarquer que juste après la fonction `tone()` se trouve une instruction `delay()`. Même si la fonction `tone()` connaît la durée pendant laquelle elle doit reproduire le son, la carte Arduino doit faire une pause pour laisser au son le temps de se terminer avant d'appeler à nouveau la fonction `tone()`. La fonction `delay()` spécifie le temps de cette pause.

La seconde manière d'utiliser `tone()` consiste à lui fournir uniquement deux arguments : le numéro de la broche utilisée et la fréquence du son à reproduire. La reproduction du son commence lors de l'appel de la fonction et ne s'arrête que lorsque la fonction `noTone()` est appelée, comme dans le programme suivant :

```
// Reproduire sur la broche 8 une tonalité de 750 Hz
tone(8, 750);
```

Copyright © 2015 Eyrolles.

```
// Le reste du code suit ici

// Stopper le son
noTone(8);
```

Les fonctions `tone()` et `noTone()` peuvent être utilisées soit dans la fonction `setup()` si tu veux que le son ne soit produit que lorsque Arduino est mis sous tension pour la première fois, soit dans la fonction `loop()` si le son doit être reproduit de façon répétitive.

Comme tu as déjà connecté le reproducteur piézoélectrique à ta carte Arduino Uno, il ne te reste qu'à télécharger le code suivant :

La version originale en anglais :

```
int piezoPin = 8;
void setup() {
  // play 3 tones when the board first starts
  tone(piezoPin, 523, 200);

  // delay is slightly longer than tone
  // so that there is silence in between the sounds
  delay(210);
  tone(piezoPin, 784, 200);
  delay(210);
  tone(piezoPin, 1047, 250);
  delay(260);
}

void loop() {
  // play 5 more tones
  tone(piezoPin, 523, 200);
  delay(210);
  tone(piezoPin, 587, 200);
  delay(210);
  tone(piezoPin, 659, 200);
  delay(210);
  tone(piezoPin, 698, 200);
  delay(210);
  tone(piezoPin, 784, 200);

  // wait 5 seconds before starting the loop over
  delay(5000);
}
```

La version francisée :

```
int brochePiezo   = 8;
```

Copyright © 2015 Eyrolles.

```
void setup() {
  // jouer 3 sons lorsque la carte est mise sous tension
  tone(brochePiezo, 523, 200);

  // le délai est légèrement supérieur à la durée de la
  // tonalité afin qu'il y ait un silence entre les sons
  delay(210);
  tone(brochePiezo, 784, 200);
  delay(210);
  tone(brochePiezo, 1047, 250);
  delay(260);
}

void loop() {
  // jouer 5 sons supplémentaires
  tone(brochePiezo, 523, 200);
  delay(210);
  tone(brochePiezo, 587, 200);
  delay(210);
  tone(brochePiezo, 659, 200);
  delay(210);
  tone(brochePiezo, 698, 200);
  delay(210);
  tone(brochePiezo, 784, 200);

  // attendre 5 secondes avant de recommencer la boucle
  delay(5000);
}
```

Lorsque Arduino exécute le code pour la première fois, tu dois entendre trois notes magiques retentir. Les cinq notes suivantes doivent ensuite se répéter indéfiniment.

EXPLORATION DU CODE

Que se passe-t-il exactement dans le sketch que tu viens de télécharger ? Sa forme est similaire à celle des autres sketches déjà étudiés. Les variables sont créées au début, tout ce qui ne se passe qu'une fois est programmé dans la fonction `setup()`, et tout ce qui est répétitif est dans `loop()`.

Le sketch contient une seule variable.

```
int piezoPin = 8 ; // version originale
```

ou

```
int brochePiezo = 8 ; // version francisée
```

Copyright © 2015 Eyrolles.

Dans les sketches précédents, la fonction `setup()` établissait le mode de fonctionnement des broches. Il n'est pas nécessaire de le faire dans ce sketch, car la seule broche qui est utilisée émet un son par l'intermédiaire de `tone()`. Cette dernière sert à démontrer la différence entre les fonctions `setup()` et `loop()`. `setup()` émet trois tonalités avant que `loop()` n'en émette cinq. Les sons de `setup()` ne sont émis qu'une seule fois, quand ta carte Arduino Uno est mise sous tension et initialisée, alors que ceux de `loop()` sont émis indéfiniment jusqu'à ce que ta carte soit mise hors tension.

La fonction `setup()` n'appelle que deux fonctions. La première est `tone()`, qui indique au microcontrôleur Arduino la fréquence sonore qu'il doit émettre, sur quelle broche il doit l'émettre et pendant quelle durée. La fonction `delay()` est ensuite appelée pour mettre en pause l'Arduino Uno en attendant que la fonction `tone()` ait terminé d'émettre le son, puis on passe à la ligne de code suivante.

La version originale en anglais :

```
// play 3 tones when the board first starts
tone(piezoPin, 523, 200);

// delay is slightly longer than tone
// so that there is silence in between the sounds
delay(210);
tone(piezoPin, 784, 200);
delay(210);
tone(piezoPin, 1047, 250);
delay(260);
```

La version francisée :

```
// jouer 3 sons lorsque la carte est mise sous tension
  tone(brochePiezo, 523, 200);

  // le délai est légèrement supérieur à la durée de la
  //tonalité afin qu'il y ait un silence entre les sons
  delay(210);
  tone(brochePiezo, 784, 200);
  delay(210);
  tone(brochePiezo, 1047, 250);
  delay(260);
```

La fonction `loop()` ressemble à `setup()` à la différence qu'elle joue cinq tonalités au lieu de trois.

La version originale en anglais :

```
// play 5 more tones
tone(piezoPin, 523, 200);
```

Copyright © 2015 Eyrolles.

```
delay(210);
tone(piezoPin, 587, 200);
delay(210);
tone(piezoPin, 659, 200);
delay(210);
tone(piezoPin, 698, 200);
delay(210);
tone(piezoPin, 784, 200);

// wait 5 seconds before starting the loop over
delay(5000);
```

La version francisée :

```
// jouer 5 sons supplémentaires
  tone(brochePiezo, 523, 200);
  delay(210);
  tone(brochePiezo, 587, 200);
  delay(210);
  tone(brochePiezo, 659, 200);
  delay(210);
  tone(brochePiezo, 698, 200);
  delay(210);
  tone(brochePiezo, 784, 200);

  // attendre 5 secondes avant de recommencer la boucle
  delay(5000);
```

Si tu sais lire une partition musicale, ouvre l'exemple Fichier > Exemples > 02.Digital > tone-Melody. Tu verras alors qu'il y a deux onglets ouverts dans l'IDE Arduino : l'un s'appelle `toneMelody`, l'autre `pitches.h`.

L'onglet `pitches.h` (*pitches* signifie « hauteurs des notes ») contient une liste de valeurs comme celles-ci :

#define NOTE_B0 31

Concentre-toi sur les termes `NOTE_B0` et `31`. Le premier est la note musicale si (B en notation anglo-saxonne) et le nombre (`31` dans cet exemple) est sa fréquence. Le numéro apparaissant après la note (`0` dans cet exemple, pour `B0`) indique l'octave. En français, on noterait :

#define NOTE_SI0 31

Tu peux utiliser ces nombres pour écrire des mélodies musicales.

Copyright © 2015 Eyrolles.

DÉFI

Le programme suivant utilise un tableau pour jouer la même séquence sonore. Crée un sketch avec ce code et télécharge-le dans ta carte Arduino Uno. Essaie ensuite de changer le tableau afin qu'il joue les sons dans l'autre ordre (des notes aiguës aux basses, au lieu des notes basses aux aiguës) :

La version originale :

```
int piezoPin = 8;
int pitches[] = {523, 587, 659, 698, 784};
int numPitches = 5;

void setup() {
  // play 3 tones when the board first starts
  tone(piezoPin, 523, 200);

  // delay is slightly longer than tone
  // so that there is silence in between the sounds
  delay(210);
  tone(piezoPin, 784, 200);
  delay(210);
  tone(piezoPin, 1047, 250);
  delay(260);
}

void loop() {
  // play 5 more tones
  int i;
  for( i=0; i<numPitches; i++){
    tone(piezoPin, pitches[i], 200);
    delay(210);
  }

  // wait 5 seconds before starting the loop over
  delay(5000);
}
```

La version francisée :

```
int brochePiezo = 8;
int hauteur[] = {523, 587, 659, 698, 784};
int nombreNotes = 5;

void setup() {
  // émettre 3 notes quand le programme démarre
  tone(brochePiezo, 523, 200);
```

Copyright © 2015 Eyrolles.

```
  // le délai est légèrement supérieur à la durée de
  // la note afin qu'il y ait un silence entre les notes
  delay(210);
  tone(brochePiezo, 784, 200);
  delay(210);
  tone(brochePiezo, 1047, 250);
  delay(260);
}

void loop() {
  // émettre 5 notes supplémentaires
  int i;
  for( i=0; i<nombreNotes; i++){
    tone(brochePiezo, hauteur[i], 200);
    delay(210);
  }

  // attendre 5 secondes avant que la boucle ne recommence
  delay(5000);
}
```

Réalisation d'un carillon à vent électronique

Un instrument de musique électronique est un instrument normal, comme une trompette ou un piano, auquel on ajoute des circuits électroniques. L'électronique permet au musicien d'utiliser un ordinateur ou un microcontrôleur pour modifier les sons que l'instrument émet normalement.

Comme tu viens de le découvrir, Arduino possède un jeu de fonctions qui permettent de générer des sons. Tu sais qu'une carte Arduino peut aussi lire des informations à l'aide de ses broches d'entrée. Tu peux donc utiliser ces informations pour émettre différents sons. Pour terminer cette aventure sonore, tu vas réaliser un carillon à vent électronique (figure 5-6) qui ne fait pas qu'émettre les sons habituels lorsqu'il est secoué par le vent : il produira également des sons provenant de ton Arduino !

Copyright © 2015 Eyrolles.

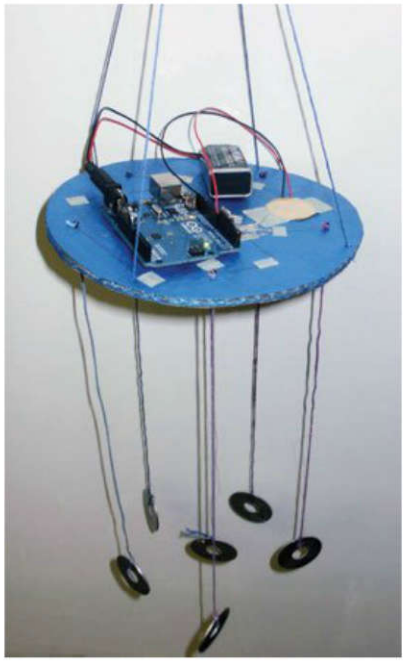


FIGURE 5-6 Le carillon à vent électronique.

Tu peux voir une vidéo (en anglais) montrant la réalisation du carillon électronique sur le site d'accompagnement www.editions-eyrolles.com/go/arduino.

Il y a beaucoup d'instruments sur lesquels tu peux ajouter de l'électronique. En voici quelques-uns sur des sites en anglais, mais en cherchant bien en ligne, tu pourras en trouver plein d'autres :

- Piano à résonance magnétique : www/eecs.qmul.ac.uk/~andrewm/mrp.html ;
- Digi Didgeridoo, un instrument à vent : http://createdigitalmusic.com/2009/12/digi-didgeridoo-augmented-wireless-digital-instrument-with-aboriginal-roots/ ;
- Augmented Beatboxing, un mélange de voix, musique et percussions : http://www.cs4fn.org/music/beatboxing.php.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

- Un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 12 fils de pontage ;
- 5 boutons poussoirs tactiles ;
- une barrette de 5 broches de connexions ;
- une pile de 9 V ;
- un adaptateur de pile 9 V à connecteur cylindrique ;
- un peu de fil fin ;
- un petit morceau de fil rigide ;
- de la cordelette, du ruban ou du fil ;
- 10 perles (en verre ou en plastique, assez grosses pour y passer la cordelette ou le ruban) ;
- 6 rondelles ou autres objets conducteurs d'électricité, faisant office de carillon ;
- du carton rigide ou du plastique servant de base ;
- du ruban adhésif opaque ;

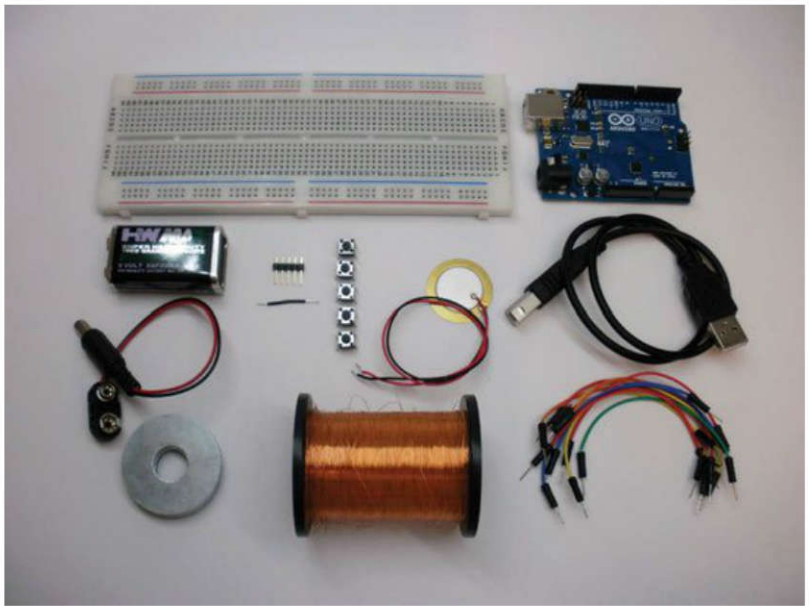


FIGURE 5-7 Les composants électroniques nécessaires à la réalisation du carillon à vent.

Copyright © 2015 Eyrolles.

- un fer à souder ;
- de la soudure ;
- des ciseaux ou un couteau de bricolage ;
- un crayon ou une perforatrice de trous ;
- un multimètre pouvant vérifier la continuité électrique.

Comprendre le circuit

Le circuit du carillon électronique n'est pas très compliqué : il combine des interrupteurs avec un haut-parleur piézoélectrique. La partie la plus amusante du circuit final utilise des matériaux que tu ne rencontres pas habituellement dans des circuits électroniques. Avant de mettre en œuvre les nouveaux éléments, il est judicieux de construire un prototype sur une plaque de montage pour t'assurer que tout fonctionne comme tu l'espères.

La figure 5-8 montre le schéma du circuit du carillon à vent. Il semble très différent du circuit final ! Les timbres y font office d'interrupteurs, bien qu'ils ne ressemblent pas à des interrupteurs normaux.

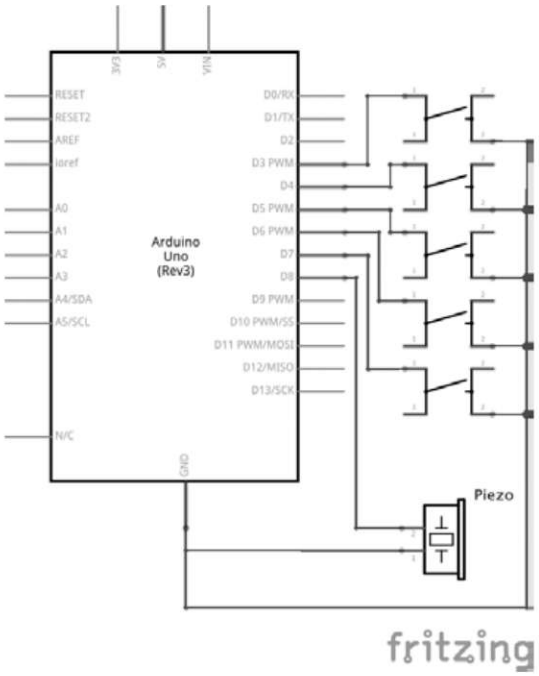


FIGURE 5-8 Le schéma du circuit du carillon à vent électronique.

Réalisation du prototype

Le circuit est constitué de cinq interrupteurs reliés à des broches numériques et un reproducteur piézoélectrique sur une autre broche numérique. Comme tu as appris comment utiliser les résistances de maintien internes de ta carte Arduino Uno lors de

Copyright © 2015 Eyrolles.

l'aventure 3, il n'est pas nécessaire d'ajouter des résistances aux interrupteurs du circuit, il n'y a donc pas beaucoup de composants à utiliser.

Construis le circuit de la figure 5-9 en suivant les étapes suivantes.

1. Positionne cinq boutons-poussoirs au-dessus de l'espace séparant les parties haute et basse de ta plaque de montage, à peu près au milieu de la plaque pour que deux broches soient insérées dans les rangées supérieures et les deux autres dans les rangées inférieures.
2. Utilise un fil de pontage pour relier la broche GND de ta carte Arduino Uno à l'une des grandes rangées en bas de la plaque. Si ta plaque de montage comporte un marquage d'une ligne bleue ou noire, ou un signe -, utilise cette rangée.
3. Utilise cinq fils de pontage pour relier la broche en bas et à droite de chaque interrupteur à la grande rangée reliée à la masse GND.
4. Utilise cinq fils de pontage pour relier la broche en haut et à droite de chaque interrupteur aux broches 3 à 7 de ta carte Arduino Uno.
5. Relie le fil rouge du haut-parleur piézoélectrique à une petite rangée inutilisée de la plaque de montage. Utilise un fil de pontage pour relier cette rangée à la broche 8 de la carte Arduino Uno.
6. Relie le fil noir du haut-parleur à la grande rangée reliée à la masse GND.

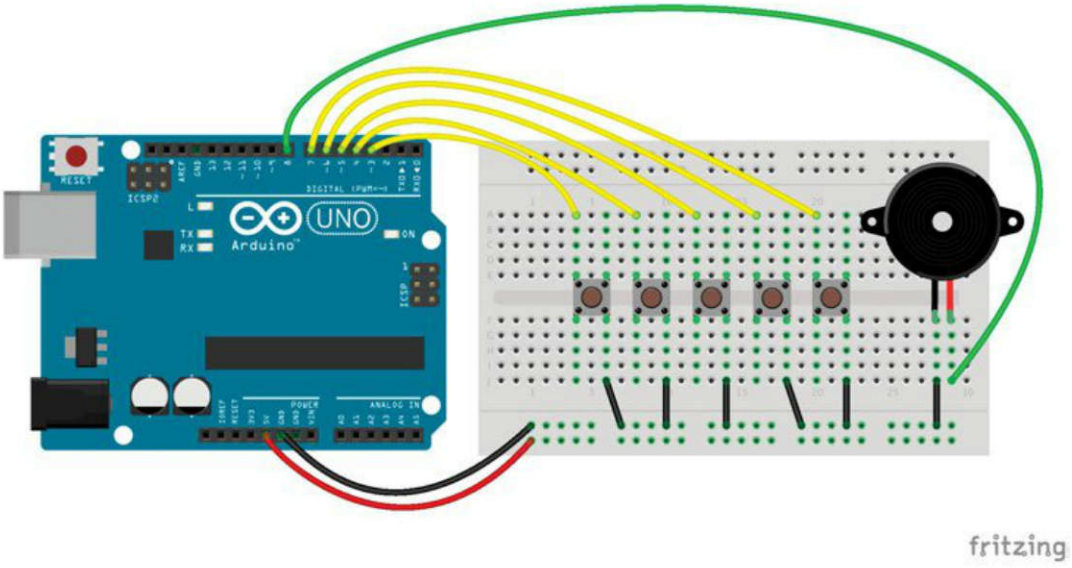


FIGURE 5-9 Le prototype du circuit sur ta plaque de montage.

Le circuit du carillon à vent est légèrement différent de celui de l'aventure 3, qui utilisait un interrupteur. À la place de boutons-poussoirs, tu utilises des éléments métalliques conducteurs (comme des rondelles ou tout autre élément que tu as choisi) reliés aux fils qui pendent de la base du carillon. Chacun de ces éléments est relié à une broche

Copyright © 2015 Eyrolles.

numérique. Une sixième partie conductrice qui est reliée à la masse GND pend au milieu des autres. Et quand le vent provoque un contact de la pièce centrale avec un des autres éléments du carillon relié à une broche numérique, l'effet est comparable à celui qui était provoqué lorsque tu appuyais sur le bouton du coffre-fort électronique de l'aventure 3.

Écriture du programme

Lance l'IDE d'Arduino, ouvre un nouveau sketch et entres-y le code suivant :

La version originale en anglais :

```
int chimes[] = {3, 4, 5, 6, 7}; // array of pins for chimes
int numChimes = 5; // total number of chimes
int piezoPin = 8; // pin for piezo

void setup(){
  int i;

  // set pinMode on all the chimes pins
  for(i=0; i<numChimes; i++) {
    pinMode(chimes[i], INPUT_PULLUP);
  }
}

void loop(){
  int i;

  for(i=0; i<numChimes; i++) {
    // read in value on pin
    int value = digitalRead(chimes[i]);

    // if LOW (meaning it has connected to ground
    if(value == LOW) {
      // play the sound
      tone(piezoPin, (100*i)+200, 30);
      delay(1);
    }
  }
}
```

La version francisée :

```
int timbres[] = {3, 4, 5, 6, 7}; // Le tableau des broches
                                 // des timbres
int nombreTimbres = 5; // Le nombre total de timbres
```

Copyright © 2015 Eyrolles.

```
int brochePiezo = 8; // La broche du haut-parleur
                     // piézoélectrique

void setup(){
  int i;

  // Appel de pinMode pour tous les timbres
  for(i=0; i<nombreTimbres; i++) {
    pinMode(timbres[i], INPUT_PULLUP);
  }
}

void loop(){
  int i;

  for(i=0; i<nombreTimbres; i++) {
    // Lecture de la valeur de la broche
    int valeur = digitalRead(timbres[i]);
    // Si niveau BAS (LOW signifiant connecté à la masse GND)
    if(valeur == LOW) {
      // Émettre le son
      tone(brochePiezo, (100*i)+200, 30);
      delay(1);
    }
  }
}
```

Télécharge ton sketch dans ta carte Arduino Uno reliée au circuit avec les interrupteurs tactiles et le haut-parleur piézoélectrique. Tu devrais entendre un son différent pour chaque bouton-poussoir sur lequel tu appuies.

Bien entendu, tu peux changer le son émis quand chaque timbre est mis en contact. Choisis toi-même ton algorithme musical.

Fabrication de ton carillon à vent

Maintenant que tu es sûr que ton sketch fonctionne correctement et que tu as réalisé ton circuit de test sur ta plaque de montage, tu es prêt à réaliser le carillon à vent. Ton carillon sera construit à partir d'une platine de base comportant six timbres qui lui sont attachés, cinq à l'extérieur et un à l'intérieur relié à la masse GND. Tu peux construire ton carillon avec n'importe quels éléments que tu auras choisis. Il est par contre important que les timbres soient conducteurs d'électricité et reliés à l'Arduino Uno. Visite le rayon quincaillerie d'une grande surface et regarde tous les éléments métalliques, parmi lesquels tu pourras choisir ce qui sera tes timbres. Les rondelles sont disponibles en différentes dimensions, mais tu pourrais aussi opter pour des écrous hexagonaux.

Copyright © 2015 Eyrolles.

EXPLORATION DU CODE

La partie principale du sketch utilise un tableau et une boucle `for` pour le parcourir, ainsi que la fonction `tone()`. La tableau `chimes()`, ou `timbres()`, contient le numéro de chaque broche reliée aux timbres (ou aux boutons poussoirs dans le cas du prototype).

La version originale :

```
int chimes[] = {3, 4, 5, 6, 7}; // array of pins
  for chimes
int numChimes = 5; // total number of chimes
int piezoPin = 8; // pin for piezo
```

La version francisée :

```
int timbres[] = {3, 4, 5, 6, 7}; // Le tableau des broches des timbres
int nombreTimbres = 5; // Le nombre total de timbres
int brochePiezo = 8; // La broche du haut-parleur piézoélectrique
```

La fonction `setup()` définit le mode de fonctionnement de chaque broche reliée aux timbres (ou boutons-poussoirs). Puisque ce sont des entrées utilisant les résistances de maintien internes de l'Arduino Uno, utilise l'argument `INPUT_PULLUP`.

La version originale :

```
int i;

// set pinMode on all the chimes pins
for(i=0; i<numChimes; i++) {
  pinMode(chimes[i], INPUT_PULLUP);
  }
```

La version francisée :

```
int i;

// Appel de pinMode pour tous les timbres
for(i=0; i<nombreTimbres; i++) {
  pinMode(timbres[i], INPUT_PULLUP);
}
```

La boucle `for` de la fonction `loop()` vérifie la valeur de chaque broche. Quand un timbre se trouve relié à la masse GND, un son est émis par `tone()`. Comme tu as utilisé une résistance de maintien interne, tu sais que la valeur du bouton-poussoir ou du timbre est au niveau `HIGH` (haut) lorsqu'il n'est pas relié à la masse GND, et au niveau `LOW` (bas) lorsqu'il est relié à la masse GND. La fréquence du son émis par `tone()` est déterminée par le timbre qui a déclenché le son.

Copyright © 2015 Eyrolles.

La version originale :

```
int i;
for(i=0; i<numChimes; i++) {
  // read in value on pin
  int value = digitalRead(chimes[i]);

  // if LOW (meaning it has connected to ground
  if(value == LOW) {
    // play the sound
    tone(piezoPin, (100*i)+200, 30);
    delay(1);
  }
}
```

La version francisée :

```
int i;
for(i=0; i<nombreTimbres; i++) {
  // Lecture de la valeur de la broche
  int valeur = digitalRead(timbres[i]);

  // Si niveau BAS (LOW signifiant connecté à
  // la masse GND)
  if(valeur == LOW) {
    // Émettre le son
    tone(brochePiezo, (100*i)+200, 30);
    delay(1);
  }
}
```

Fabriquer la base

Tu peux fabriquer la base à partir de n'importe quel matériau qui soit assez rigide et résistant pour supporter les timbres, ta carte Arduino Uno et sa pile. Du carton rigide ou du plastique conviendront parfaitement.

En premier, tu dois découper un cercle d'environ 15 cm de diamètre dans le matériau choisi. Perces-y six petits trous : les cordelettes et les fils passeront au travers. Cinq de ces trous devront être répartis de façon homogène près de la circonférence extérieure de la base ; le sixième trou devra se situer au centre.

Perce ensuite quatre trous supplémentaires pour les cordelettes qui permettront de suspendre ton carillon.

Copyright © 2015 Eyrolles.

Choisir tes timbres

Lors du choix des éléments constituant les timbres, souviens-toi de deux caractéristiques importantes : tu dois pouvoir souder les fils et ils doivent conduire l'électricité. Pour vérifier que tu peux les souder, essaie tout simplement de le faire ! Quel que soit le matériau choisi, il te faudra quelques secondes de plus pour obtenir la température de soudage parfaite par rapport à une pièce plus petite comme un morceau de fil, alors sois patient.
Pour t'assurer que le matériau choisi conduit l'électricité, tu as besoin d'un multimètre. Cet outil permet la mesure de multiples valeurs (d'où son nom bien choisi), dont les tensions et les résistances. Tu as besoin d'un multimètre permettant de mesurer la continuité électrique. Si le courant peut circuler entre ses deux fils de mesure, cela confirme la continuité. Tous les multimètres ne possèdent pas cette possibilité, fais donc attention avant d'en acheter un.

> Ne soude qu'en présence d'un adulte pouvant t'aider ! Pour être suffisamment chaud pour permettre de le souder, chaque timbre nécessite beaucoup de chaleur fournie par ton fer à souder. Un temps plus important est aussi nécessaire pour qu'il refroidisse après lui avoir soudé un fil. Sois prudent et attend au moins 5 minutes avant de saisir un élément que tu viens de souder.

Vérifie le timbre choisi en le touchant à plusieurs endroits avec les pointes des cordons de mesure de ton multimètre (voir la figure 5-10). Si celui-ci émet un bip, cela signifie que ton matériau est conducteur. Sinon, tu devras choisir un autre élément pour tes timbres.

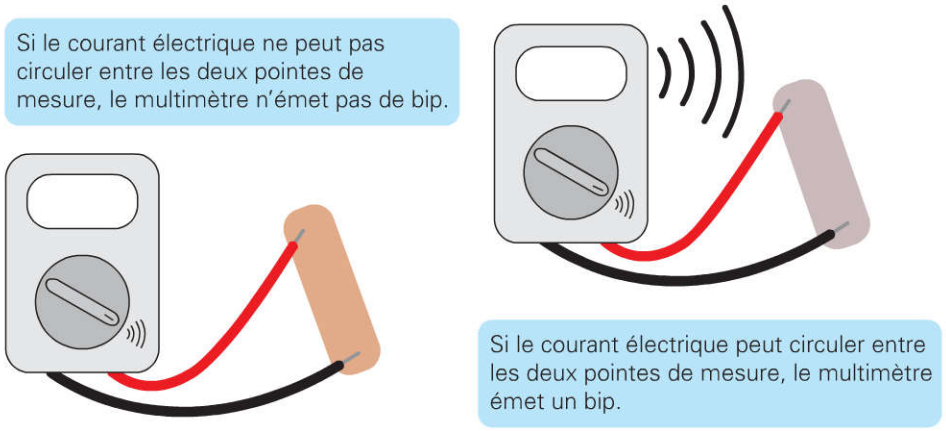


FIGURE 5-10 Le test de conductivité.

Copyright © 2015 Eyrolles.

Si tu utilises du fil très fin qui n'est pas enrobé dans une gaine de plastique isolante, il peut être protégé par une fine couche isolante. Si tu rencontres des problèmes en essayant de le souder ou s'il échoue au test de continuité, tu peux retirer la couche isolante qui l'entoure en le frottant avec du papier de verre très fin ou avec une lime à ongles.

Après avoir choisi le matériau constituant les timbres de ton carillon, soude un fil à chacun d'entre eux. Le fil doit être assez long pour supporter le timbre à la hauteur choisie tout en étant relié à ta carte Arduino Uno. Entoure le fil plusieurs fois autour de chaque timbre (voir la figure 5-11) et vérifie que tu as une bonne connexion électrique, puis soude-le au timbre.

FIGURE 5-11 Un timbre.

Le fil est bien trop faible pour soutenir le timbre lui-même. Les lustres sont toujours supportés par une solide chaîne à laquelle est attaché un fil électrique qui alimente les lampes. De la même façon, afin de fournir un meilleur soutien aux timbres de ton carillon, la section suivante t'explique comment utiliser de la cordelette ou du ruban pour les suspendre.

Attacher tes timbres

Le plus important est de se souvenir que la cordelette ou le ruban devront soutenir le poids des timbres et que le fil n'est là que pour conduire le signal électrique de l'Arduino Uno vers les timbres. Il ne doit donc soutenir aucun poids.

Noue un morceau de cordelette ou de ruban autour d'un timbre comme indiqué sur la figure 5-12. Fais passer le ruban et le fil ensemble dans leur trou, puis glisse une perle dans la cordelette ou dans le ruban (pas dans le fil). Fais un nœud dans la cordelette ou dans le ruban afin de maintenir la perle en place et ainsi d'empêcher le timbre de provoquer le retrait de la cordelette du trou (voir sur la figure 5-12).

Copyright © 2015 Eyrolles.

Fais de même pour les six timbres à suspendre.

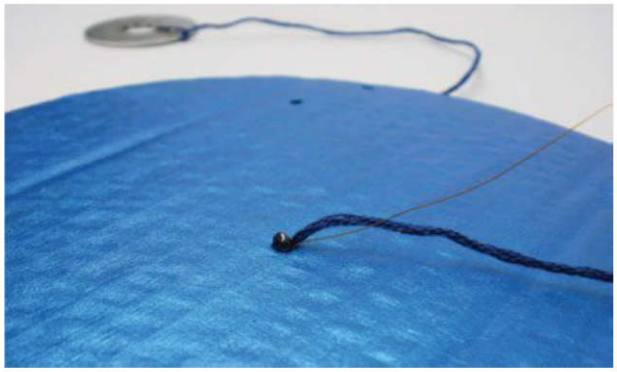


FIGURE 5-12 Un timbre attaché à la base.

Connecter ton circuit électronique

Tu as presque terminé ! Complète maintenant ces quelques étapes finales.

1. Soude un petit morceau de fil rigide au timbre suspendu au centre. Insère ce fil dans une broche GND de ta carte Arduino Uno.
2. À l'aide de ruban adhésif, fixe la carte Arduino Uno et le haut-parleur piézoélectrique sur le dessus de la base. Relie le fil noir du haut-parleur à une broche GND de la carte Arduino Uno et le fil rouge à la broche 8.
3. Soude chaque fil provenant des cinq timbres restants à une broche de la barrette de connexions. Cela demande de les souder de façon astucieuse, prends ton temps. Place ensuite la barrette de connexions sur une plaque de montage afin de maintenir les broches orientées vers le haut. Chauffe chaque broche puis étame-la avec de la soudure. Chauffe ensuite l'extrémité du fil et étame-le. Place le fil afin qu'il touche la broche de la barrette et chauffe-les ensemble à nouveau afin que la soudure qui les recouvre fonde et les relie. Répète ces opérations pour les broches restantes.
4. Insère les broches de la barrette dans les broches 3 à 7 de ta carte Arduino Uno.
5. Connecte la pile à la carte Arduino à l'aide du connecteur de pile (voir sur la figure 5-13) puis maintien la pile sur la base à l'aide de ruban adhésif.

Coupe deux morceaux de cordelette supplémentaires d'environ 60 cm de long. Ils serviront à suspendre ton carillon. Replie-les en deux et insère chacune de leurs extrémités dans les quatre trous prévus sur la base. Leurs extrémités doivent dépasser sous la base. Fais un nœud autour d'une perle glissée dans chaque cordelette pour éviter que la cordelette ne s'échappe de son trou. Suspend ton carillon et profite du son électronique généré.

Copyright © 2015 Eyrolles.

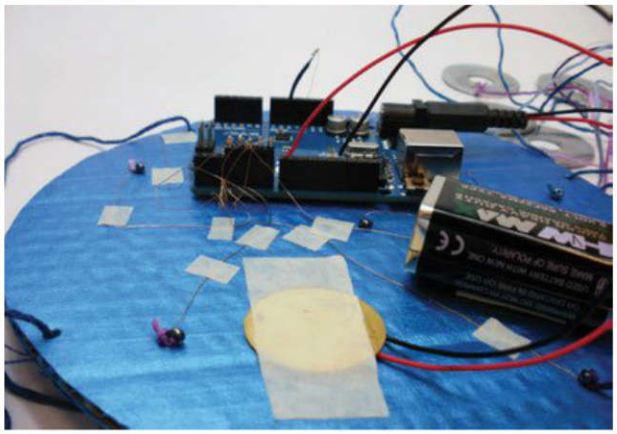


FIGURE 5-13 Le dessus de la base du carillon.

Plus d'aventures sonores

Te voilà devenu savant en effets sonores ! Grâce à tes connaissances en programmation, tu peux contrôler le son, la lumière et le mouvement à partir d'un circuit électrique. De plus, tu sais maintenant que tu n'as pas besoin d'aller dans une boutique spéciale pour acheter des matériaux pour tes circuits. En effet, puisque tu sais vérifier si un objet conduit l'électricité ou non, tu as commencé à utiliser des objets courants dans tes circuits.

Si tu souhaites en apprendre davantage sur la façon d'utiliser la fonction `tone()`, consulte la documentation Arduino à l'adresse : `http://arduino.cc/en/reference/tone`.

Tu pourrais également prendre connaissance de ces quelques autres exemples et tutoriels (en anglais) :

- `http://arduino.cc/en/Tutorial/Tone`
- `http://arduino.cc/en/Tutorial/Tone2`
- `http://arduino.cc/en/Tutorial/Tone3`
- `http://arduino.cc/en/Tutorial/Tone4`
- `http://itp.nyu.edu/physcomp/labs/labs-arduino-digital-and-analog/tone-output-using-an-arduino/`
- `https://learn.adafruit.com/adafruit-arduino-lesson-10-making-sounds/`

Si tu veux savoir comment Arduino génère des sons, regarde les pages Wikipédia relatives à la modulation de largeur d'impulsions (MLI), ou *pulse width modulation* (PWM), à l'adresse `https://fr.wikipedia.org/wiki/Modulation_de_largeur_d%27impulsion`. L'aventure 6 explique comment utiliser la modulation MLI pour contrôler la luminosité au lieu du son.

Copyright © 2015 Eyrolles.

Rappel des commandes Arduino de cette aventure	
Commande	**Description**
`[]`	Indique qu'une variable est un tableau de valeurs plutôt qu'une simple variable. Voir également `http://arduino.cc/en/Reference/Array`.
`tone`	Émet un son à une fréquence donnée. Si sa durée est également indiquée, le son est émis pendant cette durée. Dans le cas contraire, le son est émis jusqu'à l'appel de la fonction `notone()`. Voir également `http://arduino.cc/en/Reference/Tone`.
`noTone`	Arrête le son émis par `tone()`. Voir également `http://arduino.cc/en/Reference/NoTone`.

Étape franchie : Te voilà maintenant ingénieur en création sonore !

Dans la prochaine aventure

Tu utiliseras encore plus de matériaux que tu ne peux l'imaginer dans tes circuits électriques. Tu apprendras également à contrôler une LED qui change de couleur !

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Aventure 6

Ajoute des bibliothèques

IL EST TEMPS de pousser les limites de ta carte Arduino Uno. Dans les précédentes aventures, tu as utilisé les broches numériques et analogiques de ta carte. Mais sais-tu quelle est la différence entre un signal numérique et un signal analogique ? Tu as envoyé un signal de sortie sur une broche numérique dans les aventures 1 à 3, tu as lu des données sur les broches numériques dans les aventures 2 et 5, et tu as lu des données analogiques dans l'aventure 2. Mais comment fournit-on des signaux analogiques ? Patience, j'y viens. Et après avoir abordé la sortie de signaux analogiques, est-ce terminé ? Est-ce la fin de la programmation Arduino ? Pas du tout !

Tu peux pousser ton Arduino bien plus loin en utilisant les bibliothèques de code. Celles-ci te permettent d'incorporer facilement dans tes sketches des fonctions astucieuses écrites par d'autres personnes. Dans cette aventure, tu vas utiliser une bibliothèque avec laquelle tu pourras transformer (presque) n'importe quoi en un capteur sensitif. Tu vas ensuite réaliser une boule de cristal magique qui deviendra lumineuse quand tes mains l'entoureront.

Ce dont tu as besoin

Tu as besoin des éléments suivants pour la première partie de cette aventure. Les composants électroniques nécessaires sont visibles sur la figure 6-1.

Copyright © 2015 Eyrolles.

- Un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 4 fils de pontage ;
- une LED ;
- une LED RVB (rouge, vert, bleu) à cathode commune ;
- 3 résistances de 220 Ω ;
- une résistance de 10 MΩ.

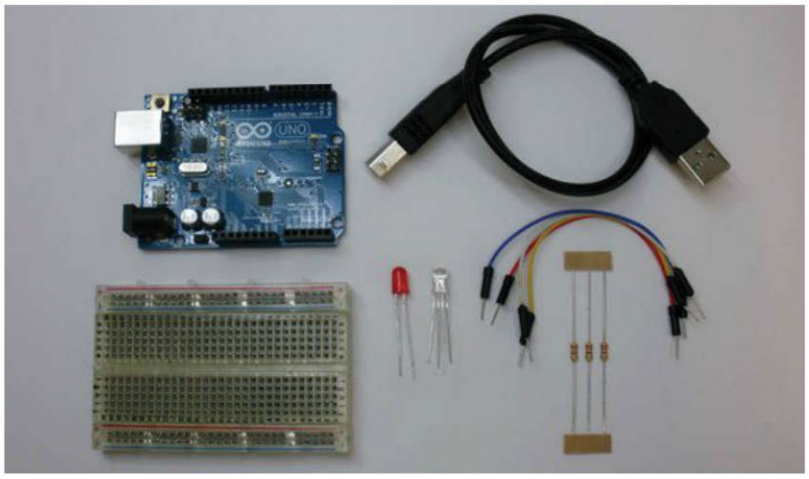


FIGURE 6-1 Les composants électroniques dont tu as besoin pour la première partie de cette aventure.

Les sorties analogiques

Si tu as suivi les aventures 1, 2 ou 4, tu sais que `digitalWrite()` peut effectuer deux actions : délivrer une tension de 5 V en utilisant le paramètre `HIGH`, ou une tension nulle de 0 V en utilisant le paramètre `LOW`. Tu peux revenir à l'aventure 3 si tu souhaites revoir l'utilisation de `digitalRead()` pour lire une donnée provenant d'un bouton-poussoir.

Tu sais également, depuis les aventures 2 et 3, que si tu souhaites mesurer la tension d'une broche qui se situe entre 0 et 5 V, tu dois utiliser `analogRead()`. Cette fonction te renvoie un nombre compris entre 0 et 1023, qui correspond à la tension d'entrée. Dès lors, on peut penser que pour sortir une tension entre 0 et 5 V, il existe probablement une fonction appelée `analogWrite()`. C'est tout à fait correct !

Mais pour commencer, voyons quelle est exactement la différence entre un signal analogique et un signal numérique.

Copyright © 2015 Eyrolles.

Un **signal numérique** est un signal électrique qui ne peut avoir que deux niveaux : ON ou OFF. Quand ce signal est représenté en électricité, il est soit de 5 V (ON), soit de 0 V (OFF). En programmation, il est `HIGH` (HAUT) s'il est ON, ou `LOW` (BAS) s'il est OFF.

> Un **signal numérique** est un signal qui ne peut être que `HIGH` (ON) ou `LOW` (OFF). Dans l'Arduino Uno, un signal `HIGH` vaut 5 V et un signal `LOW` vaut 0 V (la masse ou GND).
>
>
>

Un **signal analogique** est un signal qui peut prendre toutes les valeurs entre le niveau ON et le niveau OFF. Si c'est un signal électronique, il peut être d'une tension quelconque entre 0 et 5 V. Sa représentation en code de programmation peut être une valeur quelconque entre 0 et 1023. La figure 6-2 illustre les différences entre les signaux analogiques et numériques.

> Un **signal analogique** est un signal qui varie entre les niveaux `LOW` et `HIGH`. Sur une carte Arduino Uno, un signal analogique peut être mesuré par une valeur variant entre 0 pour le niveau 0 V de la masse ou GND et 1023 pour le niveau 5 V. Un signal analogique peut être délivré à l'aide de la valeur 0 pour 0 V et 255 pour 5 V.
>
>
>

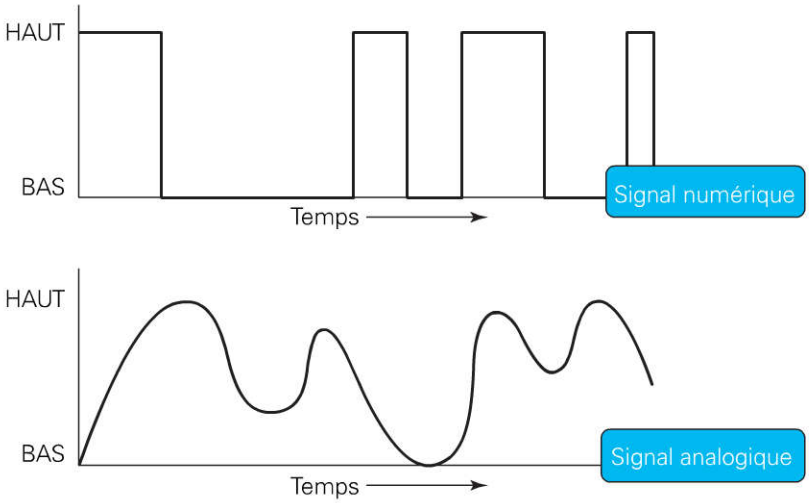


FIGURE 6-2 Signal analogique et signal numérique.

Copyright © 2015 Eyrolles.

Tu peux délivrer un signal numérique de sortie en utilisant `digitalWrite()` sur certaines broches numériques de ta carte. Tu peux aussi lire un signal numérique en utilisant `digitalRead()` sur n'importe quelle broche numérique et un signal analogique sur n'importe quelle broche analogique d'entrée. Pour délivrer un signal analogique en sortie, tu dois utiliser `analogWrite()`, mais seulement avec certaines broches numériques spéciales. Sur un microcontrôleur, délivrer une valeur comprise entre le niveau `HIGH` et le niveau `LOW` est plus délicat que de délivrer un signal numérique. C'est pour cela qu'il n'y a que certaines broches qui peuvent le faire. Elles sont précisément marquées sur ta carte Arduino avec le symbole `~`. Sur la carte Arduino Uno, il s'agit des broches 3, 5, 6, 10 et 11 (voir sur la figure 6-3).

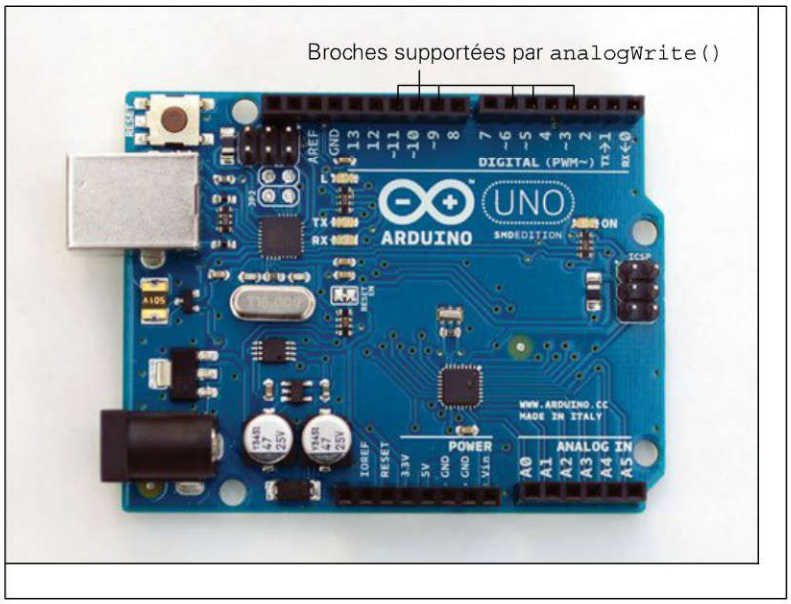


FIGURE 6-3 Les broches supportées par la fonction analogWrite().

Il est important de se rappeler que toutes les broches numériques d'une carte Arduino ne peuvent pas utiliser la fonction `analogWrite()`.

Lire et délivrer des niveaux `HIGH` et `LOW` à partir d'une broche numérique est très facile pour un microcontrôleur comme l'Arduino Uno. Toutes ses sorties et entrées peuvent le faire – et le font très bien. Tu peux même utiliser les broches analogiques pour lire ou délivrer un signal numérique quand tu es à court de broches numériques. Lire ou délivrer une tension se situant entre les niveaux `HIGH` et `LOW` est plus difficile et nécessite des fonctionnalités spéciales du microcontrôleur. C'est la raison pour laquelle `analogRead()` ne fonctionne qu'avec les broches `A0` à `A5`, et `analogWrite()` ne fonctionne qu'avec les broches identifiées par le signe `~`.

Copyright © 2015 Eyrolles.

De façon identique à `digitalWrite()`, `analogWrite()` accepte deux arguments. Le premier définit la broche qui doit être utilisée, le second la tension qui doit être délivrée. Ce deuxième argument est légèrement différent des autres. Tu ne dois pas l'exprimer en `HIGH` ou `LOW`, mais avec un nombre entre 0 et 255 : 0 pour délivrer une tension de 0 V et 255 pour une de 5 V. L'exemple de code suivant délivre en sortie, sur la broche 6, un signal approximativement égal à un tiers de 5 V ; le second argument doit donc être 83.

```
analogWrite(6, 83) ;
```

Si tu voulais délivrer une tension proche du maximum (5 V), tu devrais utiliser un nombre légèrement inférieur à 255 :

```
analogWrite(6, 249) ;
```

Contrôle de la brillance d'une LED

Pourquoi devrais-tu délivrer une tension de sortie entre 0 et 5 V ? Il y a plein de raisons à cela, mais il en est une très courante qui permet d'allumer et d'éteindre progressivement une LED.

Démarre l'IDE Arduino et ouvre le fichier Fichier > Exemples > Analog > Fading. Construis un circuit avec une LED et une résistance de limitation de courant reliée à la broche 9 (voir la figure 6-4) :

1. Utilise un fil de pontage pour relier la broche GND de ta carte Arduino Uno à une grande rangée située en bas de ta plaque de montage. Si ta plaque est marquée d'une ligne noire ou bleue ou du signe -, connecte le fil à cette rangée.
2. Insère le fil le plus court de ta LED dans la rangée reliée à la broche de la masse GND.
3. Insère le fil le plus long de ta LED dans la rangée courte libre la plus proche. Insère ensuite un fil d'une résistance dans cette rangée.
4. Insère l'autre fil de la résistance dans une autre rangée courte libre. Puis utilise un fil de pontage pour relier cette rangée à la broche 9 de ta carte Arduino Uno.

Télécharge le sketch de l'exemple et observe ce qui se passe. Tu devrais voir la LED s'allumer et s'éteindre progressivement.

Cet exemple combine `analogWrite()` et les boucles `for` que tu as déjà utilisées dans les aventures 3 à 5. La première boucle `for` augmente lentement la tension de sortie délivrée à la LED :

La version originale :

```
// fade in from min to max in increments of 5 points :
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
  // sets the value (range from 0 to 255):
```

Copyright © 2015 Eyrolles.

```
  analogWrite(ledPin, fadeValue);
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

La version francisée :

```
// Augmentation progressive du minimum au maximum
// par incréments de 5 unités :
for(int valeurSortie = 0; valeurSortie <= 255; valeurSortie
  +=5) {
  // Fixe la valeur (de 0 à 255) :
  analogWrite(brocheLed, valeurSortie);
  // Pause de 30 millisecondes permettant de voir l'effet :
  delay(30);
}
```

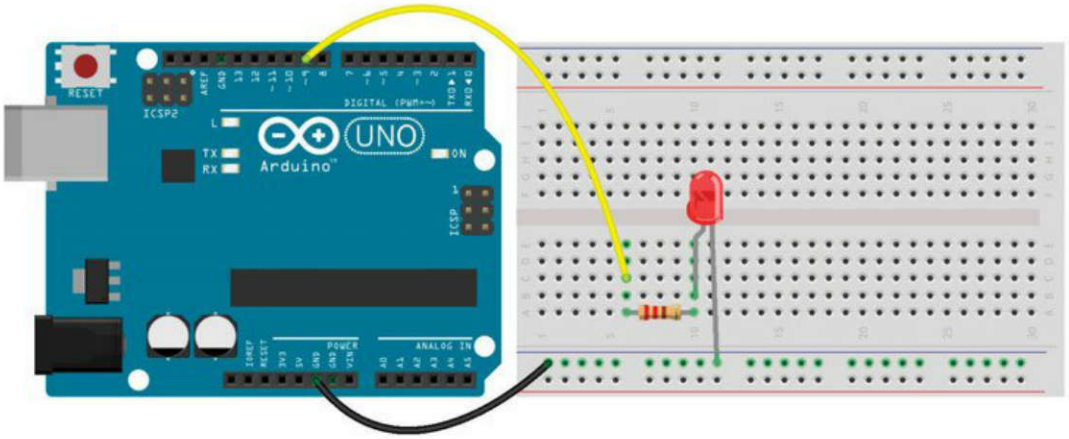


FIGURE 6-4 Le circuit permettant d'allumer et d'éteindre progressivement une LED.

La seconde boucle `for` fait le contraire en diminuant la tension délivrée :

La version originale :

```
// fade out from max to min in increments of 5 points :
for(int fadeValue = 255; fadeValue >= 0; fadeValue -=5) {
  // sets the value (range from 0 to 255):
  analogWrite(ledPin, fadeValue);
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Copyright © 2015 Eyrolles.

La version francisée :

```
// Diminution progressive du maximum au minimum
// par incréments de 5 unités :
for(int valeurSortie = 255; valeurSortie >= 0; ↩
valeurSortie -=5) {
  // Fixe la valeur (de 0 à 255) :
  analogWrite(brocheLed, valeurSortie);
  // Pause de 30 millisecondes permettant de voir l'effet :
  delay(30);
}
```

DÉFI

Augmente la vitesse à laquelle la LED s'allume et s'éteint progressivement en ajustant le temps de pause lors de l'appel de la fonction `delay()`. Ensuite, augmente la vitesse de variation en modifiant la valeur `fadeValue` (ou `valeurSortie`).

EXPLORATION DU CODE

L'utilisation de la fonction `analogWrite()` est astucieuse, car elle ne délivre pas réellement une tension fixe entre 0 et 5 V. En fait, elle utilise la **modulation de largeur d'impulsion MLI** (**PWM** ou **Pulse Width Modulation**, en anglais). Tu as peut-être remarqué qu'il est écrit PWM à côté du symbole ~ sur ta carte Arduino. Les sorties PWM délivrent un signal qui bascule alternativement entre 0 et 5 V à différentes vitesses. Cela se passe si vite qu'en éclairant ainsi une LED en utilisant une sortie PWM, tes yeux ne voient pas l'alternance des impulsions. Par contre, ils voient comme si la LED était alimentée par une tension plus faible.

La **Modulation de largeur d'impulsion** (**PWM**, ou **MLI** en français) est la technique utilisée par une carte Arduino pour générer un signal de sortie entre 0 et 5 V. Le signal bascule rapidement entre les niveaux `HIGH` et `LOW` et la tension résultante se situe entre ces deux valeurs.

La figure 6-5 montre trois exemples de signaux PWM ou MLI. Celui du haut est très faible, pratiquement nul : la LED semble presque éteinte. Celui du milieu est haut la moitié du temps : la LED est à peu près allumée, à mi-brillance. Et

Copyright © 2015 Eyrolles.

celui du bas est pratiquement haut en permanence : la LED semble éclairée comme si elle était alimentée par un signal `HIGH`.

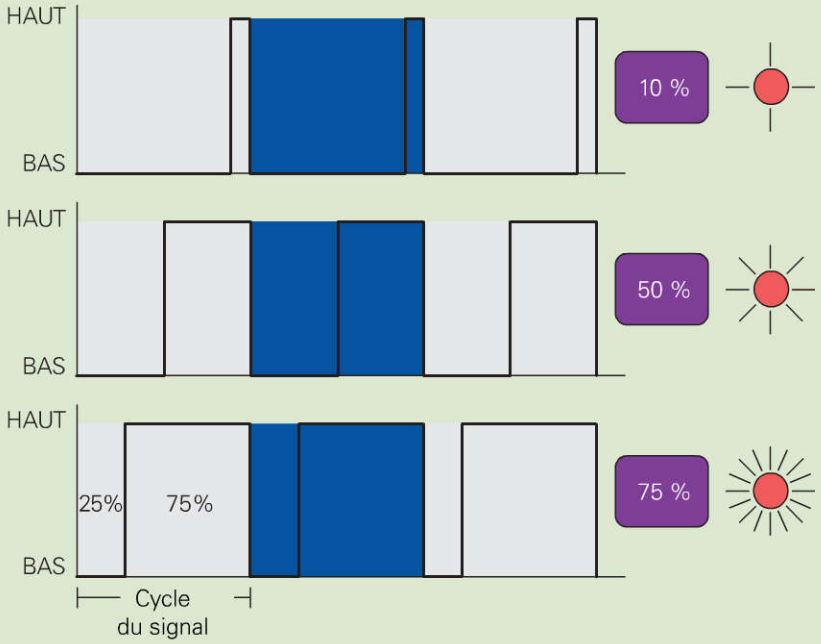


FIGURE 6-5 Exemples de modulation de largeur d'impulsion MLI (PWM).

Le rapport entre la durée pendant laquelle la sortie est au niveau `HIGH` et la durée pendant laquelle elle est au niveau `LOW` détermine la brillance apparente de la LED. Ce rapport est appelé **rapport cyclique**. Plus le temps pendant lequel le signal est HAUT est important, plus la LED paraît brillante.

Le **rapport cyclique** est le rapport entre le temps pendant lequel le signal est HAUT (`HIGH`) et le temps pendant lequel il est BAS (`LOW`) durant un cycle. En MLI (ou PWM), plus le rapport cyclique est élevé et plus la tension de sortie est élevée.

Mélange de lumières colorées

Il existe beaucoup de sortes de LED qui se présentent sous différentes formes et dimensions. Celles que tu as utilisées dans les aventures précédentes sont des LED à montage traversant – cela signifie simplement qu'elles possèdent des fils qui doivent être insérés dans les trous du circuit imprimé.

Les LED existent également en différentes couleurs. Les rouges, jaunes et vertes sont les plus courantes, mais tu peux en trouver d'autres couleurs, bleues ou orange par

Copyright © 2015 Eyrolles.

exemple. Il existe aussi des LED qui sont formées de trois diodes dans un boîtier unique ressemblant à une LED standard. Elles sont appelées **LED RGB** *(red, green, blue)* ou **RVB** (rouge, vert, bleu).

Une **LED RVB** (rouge, vert, bleu), ou **RGB** *(red, green, blue),* est une source lumineuse à LED possédant quatre fils et contenant trois LED : une rouge, une verte et une bleue. Les trois diodes ont une cathode ou une anode commune selon leur modèle.

Il existe deux sortes de LED RVB, les deux possèdent quatre fils (voir la figure 6-6). Trois des fils sont utilisés par chacune des diodes de couleur (rouge, vert et bleu). Le quatrième est commun aux trois diodes, c'est soit un fil d'**anode** (connexion positive), soit un fil de **cathode** (connexion négative). Pour les deux types, tu dois utiliser trois résistances de limitation de courant, comme tu l'aurais fait pour trois diodes séparées.

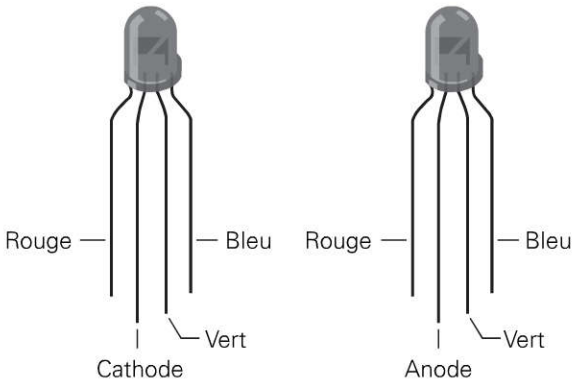


FIGURE 6-6 Les deux types de LED RVB.

L'anode est la connexion positive d'un composant polarisé, c'est-à-dire qui possède un sens de connexion précis à respecter. C'est le cas du fil le plus long d'une LED.

La **cathode** est la connexion négative d'un composant polarisé, c'est-à-dire qui possède un sens de connexion précis à respecter. C'est le cas du fil le plus court d'une LED.

Copyright © 2015 Eyrolles.

La grande différence entre les deux types de LED RVB tient au fait que l'anode commune distribue l'alimentation 5 V aux trois diodes internes, tandis que la cathode commune leur distribue la masse GND. Il est plus simple de comprendre comment le courant électrique circule dans une LED RVB à cathode commune, c'est donc ce type que tu vas utiliser.

Réalisation du câblage du circuit

Le fil le plus long de ta LED RVB est donc la cathode. C'est ce fil qui doit être connecté à la masse GND. Les trois autres fils sont reliés à la diode interne rouge, à la verte et à la bleue de ta LED RVB. Afin d'obtenir le meilleur contrôle de la luminosité, choisis trois broches de ta carte Arduino Uno pouvant être utilisées par `analogWrite()`. Les broches 9, 10 et 11 conviendront. Réalise le circuit de la figure 6-7.

1. La partie la plus délicate, quand tu utilises une LED RVB, est de déterminer quel fil contrôle chaque couleur. Le plus simple est de relier chaque fil de la LED à l'alimentation +5 V pour déterminer quelle couleur correspond à quel fil. Rappelle-toi qu'il faut utiliser une résistance de limitation de courant : tu ne dois jamais relier une LED directement à une tension de 5 V.
2. Utilise un fil de pontage pour relier la broche de masse GND de ta carte Arduino Uno. Si ta plaque est marquée d'une ligne noire ou bleue ou du signe -, connecte le fil à cette rangée.
3. Insère ta LED RVB dans ta plaque de montage afin que chaque fil soit dans sa propre rangée courte. Utilise un fil de pontage pour relier le fil le plus long de ta LED à la rangée longue connectée à la masse GND.

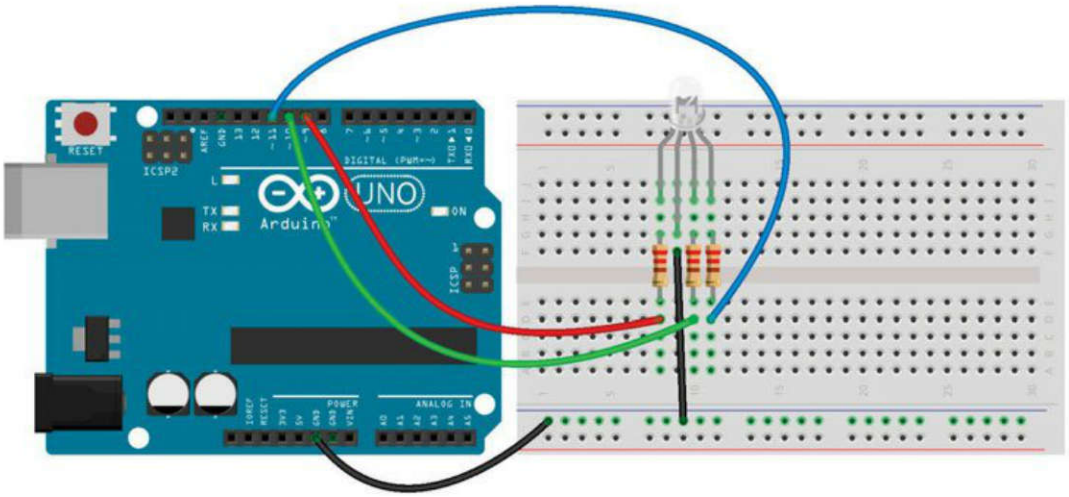


FIGURE 6-7 Le circuit pour relier ta LED RVB à ta carte Arduino Uno.

Copyright © 2015 Eyrolles.

4. Insère un des fils d'une résistance de 220 Ω dans la même rangée que chacun des fils correspondant aux couleurs de ta LED. Plie les résistances en les faisant chevaucher sur l'espace de séparation du milieu de ta plaque de montage. Insère chacun des fils libres des résistances dans sa propre rangée courte.

5. Après avoir déterminé le fil rouge de ta LED, relie la résistance qui lui correspond à la broche 9 de ta carte Arduino Uno à l'aide d'un fil de pontage. Répète cela pour la résistance correspondant à la couleur verte en la reliant à la broche 10, et pour la résistance correspondant à la couleur bleue en la reliant à la broche 11.

Écriture du programme

Le code permettant de contrôler une LED RVB ressemble à celui d'un programme qui contrôlerait trois LED. Crée un nouveau sketch dans l'IDE Arduino et écris le code suivant.

> Tu peux télécharger tous les programmes de ce livre qui ne font pas partie des exemples de l'IDE Arduino. Tu les trouveras à l'adresse www.editions-eyrolles.com/go/arduino.
>
>
>

La version originale en anglais :

```
// LED Pins
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

void setup() {
  // set pins to OUTPUT
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  // red
  for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
    analogWrite(redPin, fadeValue);
    delay(30);
  }
  for(int fadeValue = 255; fadeValue >= 0; fadeValue -=5) {
    analogWrite(redPin, fadeValue);
```

Copyright © 2015 Eyrolles.

```
  delay(30);
}

// green
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
  analogWrite(greenPin, fadeValue);
  delay(30);
}
for(int fadeValue = 255; fadeValue >= 0; fadeValue -=5) {
  analogWrite(greenPin, fadeValue);
  delay(30);
}

// blue
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
  analogWrite(bluePin, fadeValue);
  delay(30);
}
for(int fadeValue = 255; fadeValue >= 0; fadeValue -=5) {
  analogWrite(bluePin, fadeValue);
  delay(30);
}

// blue + increasing red
digitalWrite(bluePin, HIGH);
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
  analogWrite(redPin, fadeValue);
  delay(30);
}

// turn blue off again
digitalWrite(bluePin, LOW);

// green + increasing red
digitalWrite(greenPin, HIGH);
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
  analogWrite(redPin, fadeValue);
  delay(30);
}

// turn green off again
digitalWrite(greenPin, LOW);
// turn off red
digitalWrite(redPin, LOW);

// blue + increasing green
digitalWrite(bluePin, HIGH);
for(int fadeValue = 0; fadeValue <= 255; fadeValue +=5) {
```

Copyright © 2015 Eyrolles.

```
    analogWrite(greenPin, fadeValue);
    delay(30);
  }

  // turn blue off again
  digitalWrite(bluePin, LOW);
  // turn off green
  digitalWrite(greenPin, LOW);

  // turn all on to make white
  digitalWrite(redPin, HIGH);
  digitalWrite(greenPin, HIGH);
  digitalWrite(bluePin, HIGH);
  delay(2000);

  // turn all off
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
}
```

La version francisée :

```
// Les broches de la LED
int brocheRouge = 9;
int brocheVert = 10;
int brocheBleu = 11;

void setup() {
  // Définir le mode des broches en sortie OUTPUT
  pinMode(brocheRouge, OUTPUT);
  pinMode(brocheVert, OUTPUT);
  pinMode(brocheBleu, OUTPUT);
}

void loop() {
  // rouge
  for(int valeurSortie = 0; valeurSortie <= 255;
  valeurSortie +=5)
  {
    analogWrite(brocheRouge, valeurSortie);
    delay(30);
  }
  for(int valeurSortie = 255; valeurSortie >= 0;
  valeurSortie -=5)
  {
    analogWrite(brocheRouge, valeurSortie);
    delay(30);
```

Copyright © 2015 Eyrolles.

```
}

// vert
for(int valeurSortie = 0; valeurSortie <= 255;
valeurSortie +=5)
{
  analogWrite(brocheVert, valeurSortie);
  delay(30);
}
for(int valeurSortie = 255; valeurSortie >= 0;
valeurSortie -=5)
{
  analogWrite(brocheVert, valeurSortie);
  delay(30);
}

// bleu
for(int valeurSortie = 0; valeurSortie <= 255;
valeurSortie +=5)
{
  analogWrite(brocheBleu, valeurSortie);
  delay(30);
}
for(int valeurSortie = 255; valeurSortie >= 0;
valeurSortie -=5)
{
  analogWrite(brocheBleu, valeurSortie);
  delay(30);
}

// bleu + allumage progressif rouge
digitalWrite(brocheBleu, HIGH);
for(int valeurSortie = 0; valeurSortie <= 255;
valeurSortie +=5)
{
  analogWrite(brocheRouge, valeurSortie);
  delay(30);
}
// éteindre le bleu
digitalWrite(brocheBleu, LOW);

// vert + allumage progressif rouge
digitalWrite(brocheVert, HIGH);
for(int valeurSortie = 0; valeurSortie <= 255;
valeurSortie +=5)
{
  analogWrite(brocheRouge, valeurSortie);
  delay(30);
```

Copyright © 2015 Eyrolles.

```
  }
  // éteindre le vert
  digitalWrite(brocheVert, LOW);
  // éteindre le rouge
  digitalWrite(brocheRouge, LOW);

  // bleu + allumage progressif du vert
  digitalWrite(brocheBleu, HIGH);
  for(int valeurSortie = 0; valeurSortie <= 255;
  valeurSortie +=5)
  {
    analogWrite(brocheVert, valeurSortie);
    delay(30);
  }
  // éteindre le bleu
  digitalWrite(brocheBleu, LOW);
  // éteindre le vert
  digitalWrite(brocheVert, LOW);

  // allumer toutes les couleurs pour obtenir du blanc
  digitalWrite(brocheRouge, HIGH);
  digitalWrite(brocheVert, HIGH);
  digitalWrite(brocheBleu, HIGH);
  delay(2000);

  // éteindre toutes les couleurs
  digitalWrite(brocheRouge, LOW);
  digitalWrite(brocheVert, LOW);
  digitalWrite(brocheBleu, LOW);
}
```

Télécharge ce sketch dans ton Arduino Uno reliée au circuit de ta LED RVB que tu viens de réaliser. La LED devrait s'illuminer suivant une séquence répétée indéfiniment.

EXPLORATION DU CODE

Le programme réalise une séquence lumineuse qui, en premier, allume les différentes couleurs individuellement puis illumine différentes combinaisons des couleurs rouge, vertes et bleues afin de créer des couleurs résultantes. Le sketch débute par la définition des variables pour chaque broche :

```
// Les broches de la LED
int brocheRouge = 9;
int brocheVert = 10;
int brocheBleu = 11;
```

Copyright © 2015 Eyrolles.

La fonction `setup()` définit ensuite le mode de fonctionnement des broches correspondant à chaque fil de la LED :

```
// Définir le mode des broches en sortie OUTPUT
pinMode(brocheRouge, OUTPUT);
pinMode(brocheVert, OUTPUT);
pinMode(brocheBleu, OUTPUT);
```

La fonction `loop()` démarre ensuite la séquence des couleurs. En premier, chaque couleur de chaque broche est allumée puis éteinte progressivement, deux boucles `for` sont donc utilisées par broche. Le code suivant correspond à la broche rouge, mais il est répété ensuite pour les broches vertes et bleues :

```
// Rouge
for(int valeurSortie = 0; valeurSortie <= 255;
  valeurSortie +=5)
{
  analogWrite(brocheRouge, valeurSortie);
  delay(30);
}
for(int valeurSortie = 255; valeurSortie >= 0;
  valeurSortie -=5)
{
  analogWrite(brocheRouge, valeurSortie);
  delay(30);
```

Ensuite, deux broches sont allumées en même temps afin de montrer la couleur résultante. Le code suivant correspond à la combinaison du bleu avec le rouge :

```
// bleu + allumage progressif rouge
digitalWrite(brocheBleu, HIGH);
for(int valeurSortie = 0; valeurSortie <= 255;
  valeurSortie +=5) {
  analogWrite(brocheRouge, valeurSortie);
  delay(30);
}
// éteindre le bleu
digitalWrite(brocheBleu, LOW);
```

Après le mélange du bleu et du rouge, on passe au mélange du vert et du rouge, puis du vert et du bleu. Enfin, les trois couleurs sont allumées ensemble puis éteintes, et la boucle recommence indéfiniment.

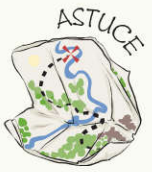
```
  // allumer toutes les couleurs pour obtenir du blanc
  digitalWrite(brocheRouge, HIGH);
  digitalWrite(brocheVert, HIGH);
```

Copyright © 2015 Eyrolles.

```
  digitalWrite(brocheBleu, HIGH);
  delay(2000);

  // éteindre toutes les couleurs
  digitalWrite(brocheRouge, LOW);
  digitalWrite(brocheVert, LOW);
  digitalWrite(brocheBleu, LOW);
```

Tu verras que tu peux différencier les LED rouge, vert et bleu à l'intérieur du boîtier de ta LED RVB quand elles sont toutes allumées. La couleur sera plus blanche que quand seulement une ou deux des couleurs sont allumées, mais elles ne seront pas totalement mélangées. La LED RVB que tu utilises est relativement peu coûteuse, mais tu pourrais dépenser un peu plus pour une LED mélangeant mieux les couleurs. Celle-ci est cependant parfaite pour commencer !

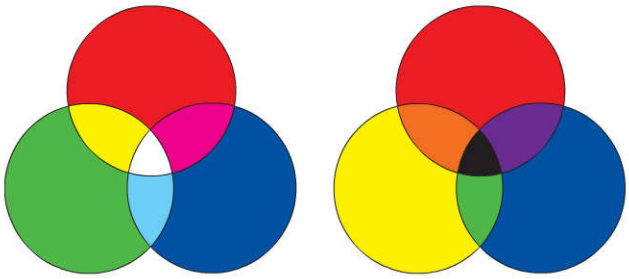


Si tu as déjà peint quelque chose, tu sais que deux couleurs peuvent être mélangées pour en créer une nouvelle. Si, par exemple, tu mélanges de la peinture bleue et de la peinture jaune, tu obtiens de la peinture verte. En mélangeant toutes les couleurs entre elles, tu obtiens du noir – ou au moins une couleur foncée proche du noir ou du marron. Il est très difficile de créer du noir sans être très précis dans les couleurs à mélanger.

Le procédé physique qui consiste à mélanger des peintures de plusieurs couleurs est appelé mélange soustractif. Le mélange de lumières colorées est, quant à lui, différent du mélange de peinture.

Si, par exemple, tu mélanges des lumières rouges et vertes, tu crées du jaune ; et si tu mélanges du rouge, du vert et du bleu, tu obtiens une lumière blanche. Regarde sur la figure 6-8 les exemples des deux types de mélanges. Tu peux chercher sur Internet des informations sur le mélange additif, pour apprendre à créer différentes couleurs avec des lumières RVB.

Copyright © 2015 Eyrolles.



Mélange additif
Lumière

Mélange soustractif
Peinture

FIGURE 6-8 Mélange de sources lumineuses et de peintures.

Les capteurs capacitifs

Sans t'en rendre compte, tu utilises tous les jours des capteurs capacitifs. Par exemple, la plupart des écrans tactiles de smartphones ou de lecteurs de musique en utilisent. Ce qui est amusant, c'est que tu peux utiliser une carte Arduino pour réaliser ton propre capteur capacitif.

La **capacité électrique,** ou **capacitance,** est une propriété qui permet de stocker des charges électriques. As-tu déjà marché dans une pièce avec de la moquette au sol puis touché quelque chose comme une voiture, un chat ou le bras d'un ami, et reçu une décharge électrique ? C'est la preuve que tu stockes des charges électriques. Un capteur capacitif détecte quand quelque chose ayant stocké des charges est proche.

La **capacité électrique** (ou **capacitance**) est une propriété permettant de stocker une charge électrique. Les composants électriques spécialement conçus pour stocker des charges sont des condensateurs. Toutefois, d'autres objets, et même les personnes, ont une capacitance.

Le code qui détecte quand quelqu'un touche ou est à proximité d'un capteur capacitif est un peu compliqué. À moins que tu ne sois intéressé par son fonctionnement détaillé, tu n'as pas à te préoccuper de ses détails – par chance, c'est là que les bibliothèques interviennent !

Ajout d'une bibliothèque

Une **bibliothèque de code** est une collection de fonctions. C'est une façon pratique d'écrire des morceaux de code que d'autres programmeurs peuvent utiliser. Cela signifie que tu n'as pas à réinventer la roue quand d'autres personnes l'ont déjà fait pour toi.

Copyright © 2015 Eyrolles.

Certaines bibliothèques sont incluses dans l'IDE Arduino, mais tu peux aussi en utiliser d'autres. Tu dois simplement les télécharger et les copier là où l'IDE Arduino les recherchera.

> Une **bibliothèque de code** est un ensemble de fonctions réutilisables que tu importes et qui peuvent être utilisées dans plusieurs sketches.

Dans le cas présent, tu vas utiliser la bibliothèque dédiée aux capteurs capacitifs. Tu peux la télécharger depuis l'adresse `http://playground.arduino.cc/Main/CapacitiveSensor`.

1. Télécharge et décompresse le dossier depuis l'adresse `https://github.com/arduino-libraries/CapacitiveSensor/zipball/master`. Le nom du dossier doit être semblable à `arduino-libraries-CapacitiveSensor-05-0-g7684dff.zip` (les lettres et chiffres à la fin du nom peuvent être différents). À l'intérieur de ce dossier, tu devrais trouver un sous-dossier nommé `exemples` ainsi que différents fichiers constituant la bibliothèque.
2. L'IDE Arduino ne recherche les bibliothèques qu'à un seul endroit : le dossier `libraries` de ton dossier d'installation. Celui-ci est le dossier `Arduino` situé dans ton dossier `Documents` ou `Mes documents` selon le système d'exploitation de ton ordinateur. On nomme parfois cet emplacement `carnet de sketches`. Copie le dossier précédemment téléchargé dans le dossier `libraries` comme indiqué sur la figure 6-9.

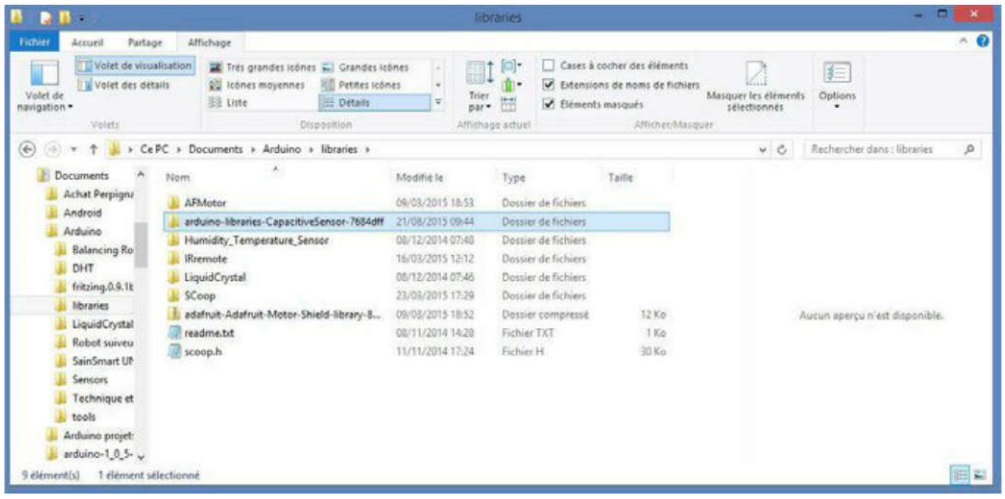


FIGURE 6-9 Copie le dossier que tu as téléchargé et décompressé dans le dossier libraries de ton dossier d'installation Arduino.

Copyright © 2015 Eyrolles.

3. Si le logiciel IDE Arduino est déjà ouvert, ferme la session et redémarre-le. Sinon, démarre simplement l'IDE.

4. Pour vérifier que la bibliothèque est installée correctement, utilise le menu Croquis > Importer bibliothèque et regarde si `CapacitiveSensor` apparaît dans la liste comme sur la figure 6-10. Regarde également le menu Fichier > Exemples, tu devrais y trouver `CapacitiveSensor`.

Si tu ne vois pas les exemples ou la bibliothèque dans les listes des menus, redémarre l'IDE Arduino. Si cela ne les fait pas apparaître, reviens en arrière et répète les étapes d'installation pour être sûr que tu as correctement renommé et situé le dossier téléchargé au bon emplacement.

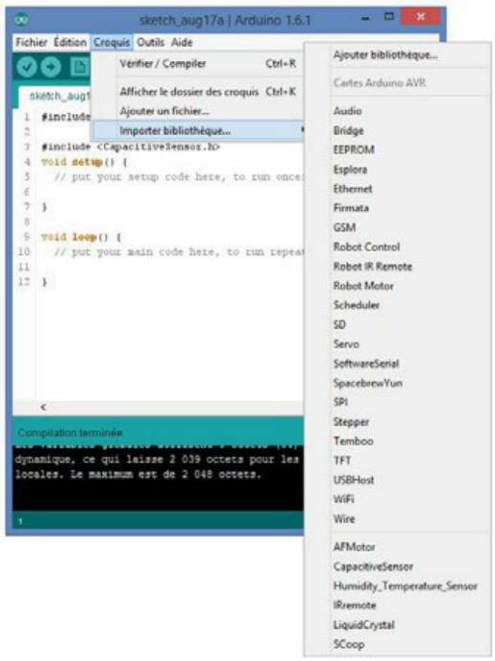


FIGURE 6-10 Vérifie la présence de la bibliothèque et des exemples dans tes menus.

Réalisation du câblage de ton circuit

Le circuit d'un capteur capacitif utilise deux broches. La première envoie un signal à une antenne, la seconde détecte le signal reçu en retour. L'antenne peut être n'importe quel objet conducteur de l'électricité. Du papier aluminium fonctionne parfaitement, mais tu peux aussi utiliser tout simplement un morceau de fil électrique auquel rien n'est relié. Quand un objet capacitif (par exemple ton doigt) approche de l'antenne, le signal reçu est modifié selon sa distance. La bibliothèque retourne un nombre correspondant à cette distance.

Copyright © 2015 Eyrolles.

Tu peux souder un fil à un trombone métallique (assure-toi qu'il n'est pas recouvert de plastique, sinon, dénude-le à l'aide de ta pince à dénuder). Tu peux alors utiliser ce trombone pour maintenir un morceau de papier aluminium ou un autre objet et le relier à ta plaque de montage avec le fil.

Le circuit du capteur capacitif utilise une résistance entre l'antenne et la broche envoyant le signal. Tu dois utiliser une résistance de valeur élevée – au moins 10 MΩ (c'est-à-dire 10 millions d'ohms) ! Construis le circuit de la figure 6-11.

1. Insère un fil de la résistance de 10 MΩ au milieu de ta carte de montage. Insère son autre fil dans une autre rangée.
2. Utilise un fil de pontage pour relier un des côtés de la résistance à la broche 2 de ta carte Arduino Uno et un second fil de pontage pour connecter l'autre extrémité de cette résistance à la broche 4 de ta carte.
3. Connecte un autre fil de pontage à la rangée de la plaque de montage reliée à la broche 2. Ne connecte pas l'autre extrémité de ce fil. C'est l'antenne de ton capteur capacitif.

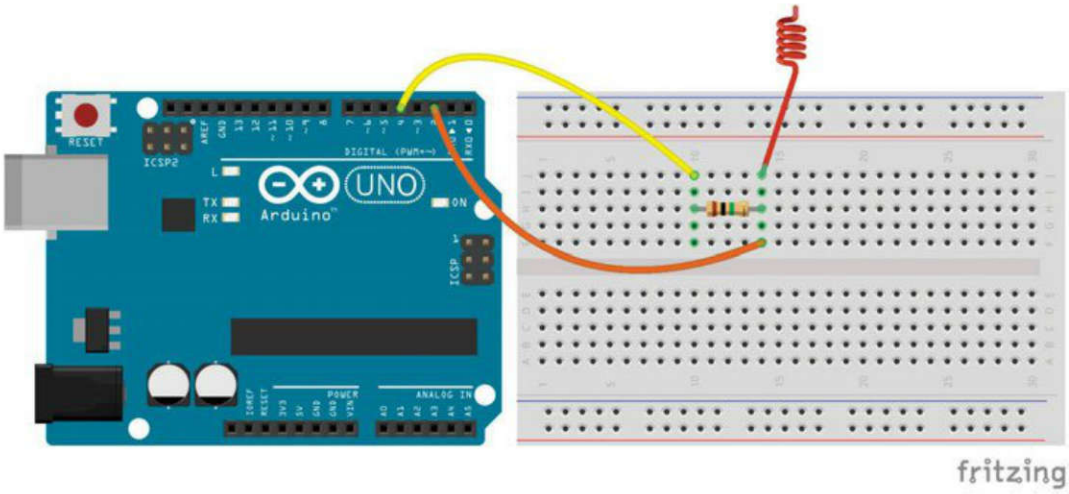


FIGURE 6-11 Le circuit du capteur capacitif.

Écriture du programme

Quand tu utilises une bibliothèque, tu dois d'abord indiquer laquelle à l'IDE Arduino. Pour cela, tu dois ajouter dans le code l'instruction `#include` suivie du nom du fichier contenant la bibliothèque, comme ceci :

```
#include <CapacitiveSensor.h>
```

Copyright © 2015 Eyrolles.

L'extension du fichier est `.h` et les symboles `<` et `>` signifient que la bibliothèque se trouve à l'emplacement prévu par l'IDE Arduino : le dossier `libraries` de ton carnet de sketches (qui est le dossier `Arduino` de tes documents).

La bibliothèque définit un objet appelé `CapacitiveSensor`. Cet objet prend en charge tous les détails pratiques permettant l'interaction avec le capteur. Tu dois créer une nouvelle variable dont le type est `CapacitiveSensor`. Seulement deux arguments sont nécessaires : le numéro de la broche émettant le signal et le numéro de celle qui est reliée au capteur :

```
CapacitiveSensor CapteurTactile =
  CapacitiveSensor(outputPin, sensorPin) ;
```

Pour lire les données en provenance du capteur, la fonction `capacitiveSensor()` est appelée. On appelle la fonction `capteurTactile.capacitiveSensor()`, car elle appartient à la variable `capteurTactile` de type `CapacitiveSensor`.

```
long valeurCapteur = capteurTactile.capacitiveSensor(30)
```

La valeur retournée est mémorisée dans une variable de type **long**. L'argument d'appel de cette fonction (30 ici) représente le nombre de lectures faites. Plus ce nombre est élevé et plus le fonctionnement du capteur est fiable, au détriment de la durée de lecture. La valeur retournée est un nombre arbitraire indiquant si le capteur est inactif (valeur basse) ou actif (valeur élevée). Le type `long` est similaire au type `int`, sauf que les variables de ce type peuvent mémoriser des valeurs beaucoup plus petites et beaucoup plus grandes que celles des variables de type `int`. Il est préférable d'utiliser le type `long` dans notre cas.

Le type de donnée **long** peut mémoriser des nombres entiers compris entre -2 147 483 648 et 2 147 483 647.

En utilisant toutes ces instructions, voici le sketch résultant.

La version originale :

```
#include <CapacitiveSensor.h>

// capSense pins
int sensorPin = 2;
int outputPin = 4;
CapacitiveSensor handSensor =
```

Copyright © 2015 Eyrolles.

```
  CapacitiveSensor(outputPin,sensorPin);

void setup() {
  // begin serial communication
  Serial.begin(9600);
}

void loop() {
  // read in the value from the sensor
  long sensorValue = handSensor.capacitiveSensor(30);
  // print the value
  Serial.println(sensorValue);
  // wait for a short while before continuing
  delay(10);
}
```

La version francisée :

```
#include <CapacitiveSensor.h>

// Les broches utilisées par le capteur
int brocheCapteur = 2;
int brocheSignal = 4;
CapacitiveSensor capteurTactile = ↩
CapacitiveSensor(brocheSignal,brocheCapteur);

void setup() {
  // initialiser une communication série
  Serial.begin(9600);
}

void loop() {
  // lire les données retournées par le capteur
  long valeurCapteur = capteurTactile.capacitiveSensor(30);
  // afficher la valeur
  Serial.println(valeurCapteur);
  // faire une courte pause avant de continuer
  delay(10);
}
```

Télécharge ce sketch dans ta carte Arduino Uno et ouvre le moniteur série de l'IDE Arduino sur ton ordinateur. Touche l'extrémité du fil de pontage jouant le rôle d'antenne et regarde les valeurs qui sont affichées.

Copyright © 2015 Eyrolles.

EXPLORATION DU CODE

Ce sketch n'est pas très long, mais regardons-le tout de même en détails. Au début, les variables du sketch sont déclarées. Deux broches sont nécessaires au capteur capacitif, celle qui est reliée à la résistance et celle reliée à l'antenne :

```
// Les broches utilisées par le capteur
int brocheCapteur = 2;
int brocheSignal = 4;
```

Dans la fonction `setup()`, la seule action effectuée est le démarrage d'une communication série :

```
// initialiser une communication série
Serial.begin(9600);
```

La fonction `loop()` est également courte. La valeur retournée par le capteur capacitif est mémorisée dans la variable `valeurCapteur`, puis elle est affichée dans le moniteur série. La fonction `delay()` est ensuite appelée pour faire une pause avant que le programme ne se répète ;

```
// lire les données retournées par le capteur
long valeurCapteur = capteurTactile.capacitiveSensor(30);
// afficher la valeur
Serial.println(valeurCapteur);
// faire une courte pause avant de continuer
delay(10);
```

Que se passe-t-il si tu ne touches pas la partie métallique du fil de pontage servant d'antenne, mais la partie isolée en plastique ?

Copyright © 2015 Eyrolles.

Essaie d'ajouter en série des résistances de 10 MΩ afin d'augmenter la sensibilité de ton capteur. Cela lui permettra de détecter l'approche de ta main à une distance supérieure. Ajoute-les en série, c'est-à-dire bout à bout. Quand tu relies des résistances de cette manière, la valeur totale de la résistance obtenue est la somme des valeurs de résistance de chacune d'elles. C'est pratique et utile, car il est presque impossible d'acheter des résistances de valeurs supérieures à 10 MΩ. Tu peux ainsi mettre plusieurs résistances en série qui apparaîtront comme une résistance de forte valeur dans ton circuit.

Fabrication de ta boule de cristal

Les capteurs capacitifs peuvent paraître magiques, même si tu sais que ce n'est qu'un processus électrique. Ainsi, profites-en pour accentuer ce point en créant une boule de cristal qui s'illuminera par magie lorsque tu approcheras ta main en un mouvement de vague, comme sur la figure 6-12.

FIGURE 6-12 Une boule de cristal sensitive.

Tu peux fabriquer la tienne en papier mâché placé sur un ballon. C'est un excellent moyen de réaliser une sphère semi-transparente qui s'illuminera joliment grâce à ta LED RVB placée à l'intérieur.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

Tu as besoin des éléments suivants pour réaliser ta boule de cristal. Les composants électroniques nécessaires sont présentés sur la figure 6-13 :

- un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- 8 fils de pontage ;
- du fil de câblage rigide ;
- 3 résistances de 220 Ω ;
- 4 résistances de 10 MΩ ;
- une LED RVB à cathode commune ;
- un trombone ;
- une pile de 9 V ;
- un adaptateur pour la pile 9 V avec connecteur cylindrique ;
- 3 feuilles de papier mâché ;
- un ballon gonflable ;
- du carton mince ou du papier épais ;
- de la feuille d'aluminium ;
- un fer à souder ;
- de la soudure ;
- des ciseaux ou un couteau de bricolage ;
- de la colle blanche d'écolier ;
- un pinceau.

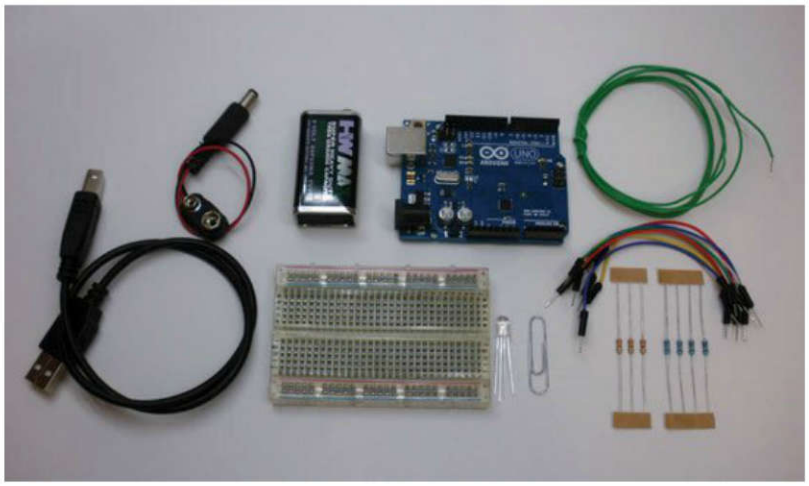


FIGURE 6-13 Les composants électroniques dont tu as besoin pour réaliser ta boule de cristal.

Comprendre le circuit

Le circuit de la boule de cristal est une combinaison du circuit de la LED RVB et de celui du capteur capacitif que tu as réalisés précédemment dans cette aventure. Il est plus facile de comprendre un circuit en regardant son schéma plutôt que sa réalisation sur une plaque de montage. Regarde donc la figure 6-14.

Copyright © 2015 Eyrolles.

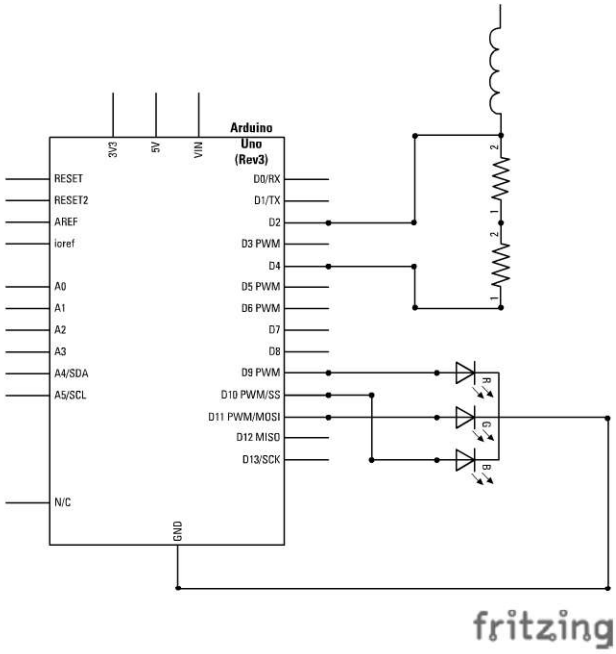


FIGURE 6-14 Le schéma du circuit de la boule de cristal.

Réalisation du prototype sur plaque de montage

Si tu as complété toutes les aventures précédentes de ce livre, tu es maintenant un technicien expert. Tu sais donc combien il est important de vérifier ton circuit sur une plaque de montage, avant de commencer à en souder les composants. Je tiens néanmoins à te rappeler de le faire, au cas où tu voudrais te précipiter. Réalise le circuit de la figure 6-15 en suivant ces étapes.

1. Utilise un fil de pontage pour relier la broche GND de ta carte Arduino Uno à une rangée longue en bas de ta plaque de montage. Si ta plaque est marquée d'une ligne noire ou bleue ou par le signe -, relie la broche GND à cette rangée.
2. Place ta LED RVB sur ta plaque de montage afin que chaque fil soit inséré dans sa propre rangée courte. Utilise un fil de pontage pour relier le fil le plus long de la LED à la rangée longue connectée à la broche de masse GND.
3. Insère un des fils d'une résistance de 220 Ω dans chaque rangée où un fil de couleur de la LED est inséré. Plie la résistance de façon à ce qu'elle se place au-dessus de la séparation de la plaque de montage, puis insère son autre fil dans sa propre rangée.
4. Repère le fil de la couleur rouge et relie la résistance qui lui est associée à la broche 9 de ta carte Arduino Uno à l'aide d'un fil de pontage. Répète cette opération afin de connecter la résistance reliée au fil de la couleur verte à la broche 10, puis à nouveau pour connecter la résistance reliée au fil de la couleur bleue à la broche 11.

Copyright © 2015 Eyrolles.

5. À l'autre extrémité de ta plaque de montage, insère les fils d'une résistance de 10 MΩ dans deux rangées différentes, au milieu de ta plaque de montage.
6. Insère un des fils d'une seconde résistance de 10 MΩ dans une des rangées utilisées par la première résistance. Insère l'autre fil dans une rangée disponible.
7. Connecte un fil de pontage entre la broche 2 et l'une des résistances (ne le relie pas à la rangée commune aux deux résistances). Relie un autre fil de pontage à cette même rangée, il servira d'antenne pour ton circuit prototype.
8. Relie un fil de pontage entre la broche 4 et le fil de la résistance qui n'est pas encore relié.

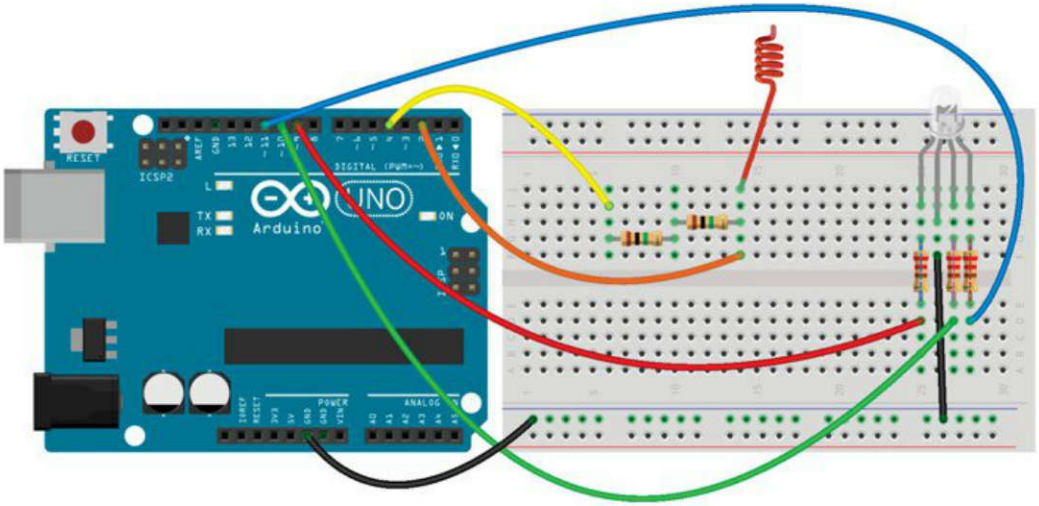


FIGURE 6-15 Le circuit de ton prototype sur ta plaque de montage.

Après avoir réalisé le support de ta boule de cristal avec du papier et de feuille d'aluminium, utilise le circuit prototype pour déterminer combien de résistances sont nécessaires pour ton capteur capacitif. Il t'en faudra une à quatre. Tu peux ajouter plus de résistances aux deux présentes sur ta plaque de montage en les mettant bout à bout (en série) du côté de leur connexion avec ta carte Arduino Uno.

Écriture du programme

Les premières lignes du programme sont un mélange des sketches de la LED RVB et du capteur capacitif étudiés précédemment dans cette aventure. Dans ce code, les variables pour les broches et le capteur sont déclarées. Tu construiras le support de ta boule dans la section suivante. La boule sera couverte de feuille d'aluminium constituant l'antenne de ton circuit. Pour l'instant, commence d'abord par créer un nouveau sketch dans l'IDE Arduino et entres-y les instructions suivantes :

Copyright © 2015 Eyrolles.

La version originale :

```
#include <CapacitiveSensor.h>

// LED pins
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

// capSense pins
int sensorPin = 2;
int outputPin = 4;

// touch threshold
int threshold = 1000;

CapacitiveSensor handSensor = CapacitiveSensor(outputPin,
  sensorPin);

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  long sensorValue = handSensor.capacitiveSensor(30);
  Serial.println(sensorValue);

  // if above the threshold
  if(sensorValue > threshold) {
    // calculate color value based on sensor reading
    int redValue = map(sensorValue, threshold, 90000, 0, 255);
    int greenValue = map(sensorValue, threshold, 20000, 0,
  255);
    int blueValue = map(sensorValue, threshold, 30000, 0,
  255);
    // turn on led
    analogWrite(redPin, redValue);
    analogWrite(greenPin, greenValue);
    analogWrite(bluePin, blueValue);
  }
    else {
    // otherwise turn off led
    digitalWrite(redPin, LOW);
```

Copyright © 2015 Eyrolles.

```
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
  }
}
```

La version francisée :

```
#include <CapacitiveSensor.h>

// Les broches des LED
int brocheRouge = 9;
int brocheVert = 10;
int brocheBleu = 11;

// Les broches du capteur capacitif
int brocheCapteur = 2;
int brocheSignal = 4;

// Le seuil de déclenchement du capteur
int seuil = 1000;

CapacitiveSensor capteurTactile =
  CapacitiveSensor(brocheSignal,  brocheCapteur);

void setup() {
  pinMode(brocheRouge, OUTPUT);
  pinMode(brocheVert, OUTPUT);
  pinMode(brocheBleu, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  long valeurCapteur = capteurTactile.capacitiveSensor(30);
  Serial.println(valeurCapteur);

  // Cas où la valeur est supérieure au seuil
  if(valeurCapteur > seuil) {
    // Déterminer la couleur en fonction de la valeur du
    // capteur
    int valeurRouge = map(valeurCapteur, seuil, 90 000, 0,   255);
    int valeurVert = map(valeurCapteur, seuil, 20 000, 0,   255);
    int valeurBleu = map(valeurCapteur, seuil, 30 000, 0,   255);
    // Allumage de la couleur rouge, puis verte, puis bleue
    analogWrite(brocheRouge, valeurRouge);
    analogWrite(brocheVert, valeurVert);
    analogWrite(brocheBleu, valeurBleu);
  }
```

Copyright © 2015 Eyrolles.

```
    else{
    // si la valeur est inférieure au seuil, éteindre les LED
    digitalWrite(brocheRouge, LOW);
    digitalWrite(brocheVert, LOW);
    digitalWrite(brocheBleu, LOW);
  }
}
```

Télécharge ce sketch dans ta carte Arduino Uno, puis touche le fil de pontage faisant office antenne. Tu devrais voir la LED s'allumer et changer de couleur. Ouvre le moniteur série de l'IDE Arduino pour observer les valeurs reçues par le capteur tactile.

EXPLORATION DU CODE

La variable `seuil` est importante, car c'est elle qui te permet d'ajuster les interactions avec ta boule de cristal. Cette variable tient un rôle essentiel dans la fonction `loop()`. Elle détermine la sensibilité de ta boule de cristal en maintenant la LED éteinte jusqu'à ce qu'une valeur retournée par le capteur soit suffisamment élevée :

```
// Le seuil de déclenchement du capteur
int seuil = 1000 ;
```

Observe la valeur retournée dans le moniteur série pour ajuster la valeur du seuil de déclenchement. Tu devras l'ajuster plusieurs fois, car ton antenne sera différente dans le prototype et avec la boule de cristal.

Dans la fonction `loop()`, `seuil` détermine le moment où la lumière s'allume :

```
// Cas où la valeur est supérieure au seuil
  if(valeurCapteur > seuil) {;
```

À l'intérieur de cette instruction `if`, il y a trois lignes de code dans lesquelles on utilise une nouvelle fonction : `map()`. Cette fonction prend un nombre dans des limites données et le remplace par un nombre contenu dans des limites différentes. Elle est très utile car les valeurs retournées par le capteur sont très grandes, alors qu'`analogWrite()` ne peut pas accepter de valeur supérieure à `255`. La fonction `map()` réétalonne la valeur renvoyée par le capteur en la situant dans les limites acceptables. La figure 6-16 explique son fonctionnement.

Le premier argument de `map()` est le nombre à réétalonner. Le second et le troisième argument sont les limites inférieure et supérieure de ce nombre. Et, le quatrième et le cinquième sont les limites inférieure et supérieure que la valeur renvoyée doit respecter :

```
int valeurRouge = map(valeurCapteur, seuil,
  90 000, 0, 255);
```

Copyright © 2015 Eyrolles.

```
int valeurVert = map(valeurCapteur, seuil,
  20 000, 0, 255);
int valeurBleu = map(valeurCapteur, seuil,
  30 000, 0, 255); {;
```

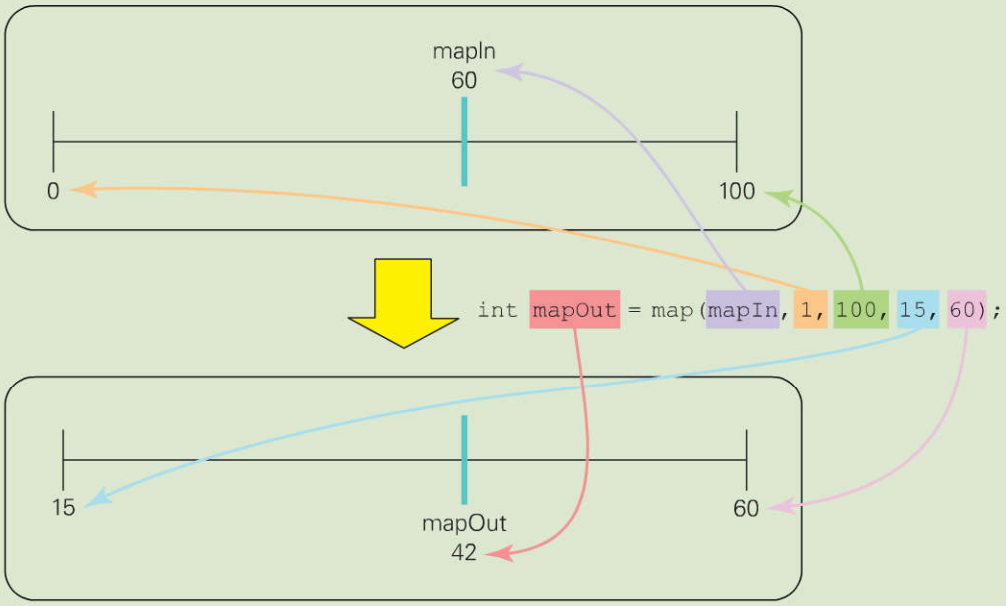


FIGURE 6-16 Réétalonnage d'une valeur dans des limites différentes.

Dans le sketch, la valeur de la limite inférieure est différente pour chaque couleur. Ceci permet à chaque couleur de répondre différemment pour une même valeur du capteur. Par exemple, une valeur de 3 000 retourne 6 pour `valeurRouge`, 28 pour `valeurVert` et 18 pour `valeurBleu`. Tu peux essayer des valeurs différentes dans les trois fonctions `map()` afin d'obtenir les couleurs que tu préfères.

Construction de ta boule de cristal

Maintenant, tu vas fabriquer ta boule de cristal en utilisant du papier mâché pour créer une mince sphère qui laissera la lumière la traverser. La base sur laquelle reposera ta boule sera constituée de papier cartonné ou de papier épais recouvert d'une feuille d'aluminium. Elle servira de capteur capacitif. De plus, l'électronique y logera parfaitement et, si tu utilises une pile pour l'alimenter, l'illusion magique sera incroyable.

Fabrique ta boule en papier mâché

La boule de cristal sera faite de papier mâché dont la forme sera obtenue à l'aide d'un ballon. Utilise du papier tissu plutôt que du papier épais afin que la lumière de la LED RVB puisse passer au travers.

Copyright © 2015 Eyrolles.

1. Gonfle un ballon à la dimension souhaitée pour ta boule de cristal. Souviens-toi que si tu veux placer ta carte Arduino Uno dans la base, ton ballon doit être assez gros pour se placer dans la base contenant la carte Arduino Uno.
2. Détrempe une cuillérée à café de colle blanche (environ 25 ml) avec une demi-cuillérée à café d'eau (15 à 25 ml).
3. Découpe trois feuilles de papier tissu d'environ 2,5 cm au carré.
4. Utilise un pinceau pour appliquer le mélange de colle au sommet du ballon opposé au nœud.
5. Place un morceau de papier tissu sur la surface encollée et applique de la colle dessus.
6. Répète les étapes 4 et 5 en recouvrant le ballon de morceaux de papier tissu en descendant vers le nœud, jusqu'à ce que le ballon soit recouvert aux deux tiers.
7. Laisse le ballon sécher au moins une demi-journée, ou une nuit.
8. Répète ces actions pour poser une deuxième couche de papier tissu. Tu devras en poser jusqu'à trois couches si tu veux que le papier séché soit assez solide pour conserver sa forme une fois que tu auras retiré le ballon.
9. Quand le papier mâché est bien sec et assez rigide pour conserver sa forme, perce le ballon. Retire le plastique du ballon avec précaution à l'intérieur de la sphère de papier mâché. Si nécessaire, coupe le bord du papier mâché pour obtenir un « cristal » avec un bord net. Tu devrais alors obtenir un objet ressemblant à celui de la figure 6-17.
10. Coupe une bande de papier cartonné ou de papier épais d'environ 5 cm de large. Recouvre-la de papier aluminium.
11. Plie le carton recouvert d'aluminium pour obtenir un anneau juste assez grand pour supporter la boule et pour loger ta carte Arduino Uno (voir la figure 6-18). Termine l'anneau en l'agrafant ou en le collant avec du ruban adhésif.
12. Attache ta boule avec de la colle ou du ruban adhésif.

FIGURE 6-17 Ta boule de cristal en papier mâché.

FIGURE 6-18 Le support recouvert de feuille d'aluminium.

Copyright © 2015 Eyrolles.

Soude le circuit électronique

Il n'y a pas beaucoup de composants dans ce circuit, mais tu dois tout de même en réaliser d'abord un prototype, sinon tu ne sauras pas combien de résistances tu dois utiliser pour ton capteur.

Soude une résistance de 220 Ω à chacun des trois fils courts de ta LED RVB. Coupe un morceau de fil de câblage rigide de longueur identique à la résistance et soude-le à la cathode de ta LED RVB (voir la figure 6-19).

Souder peut être dangereux, car le fer à souder est très chaud. Ne soude qu'en présence d'un adulte qui pourra t'aider.

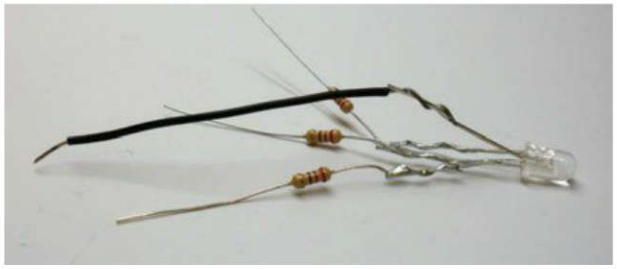


FIGURE 6-19 Soudure du circuit de la LED.

Soude ensemble la quantité de résistances de 10 MΩ dont tu as besoin pour ton circuit.

Coupe un morceau de fil de câblage rigide d'environ 8 cm. Soude une de ses extrémités à un trombone et l'autre extrémité à la dernière résistance de 10 MΩ de ta chaîne de résistances (voir la figure 6-20).

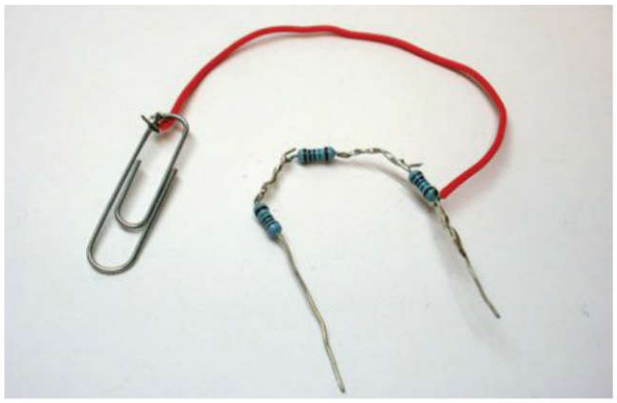


FIGURE 6-20 Soudure du circuit du capteur.

Copyright © 2015 Eyrolles.

Connecte ton électronique

Insère l'extrémité non soudée du fil de câblage des résistances de ton capteur dans la broche 4, puis le fil relié au trombone dans la broche 2. Insère les résistances reliées aux fils des couleurs rouge, vert et bleu de ta LED RVB dans les broches 9, 10 et 11 (voir la figure 6-21). Attache le trombone à la feuille d'aluminium du support de ta boule de cristal.

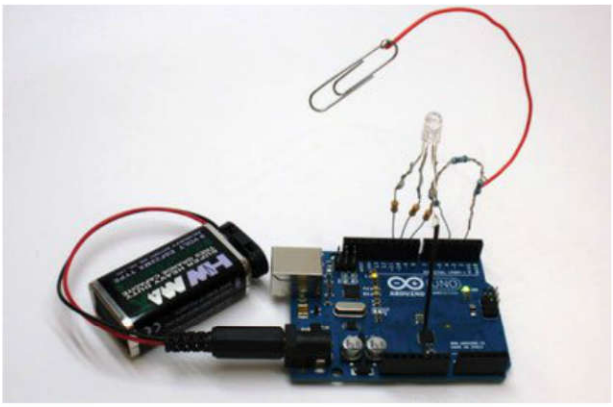


FIGURE 6-21 Le circuit de la boule de cristal terminé.

Mets sous tension ta carte Arduino Uno à l'aide d'une pile de 9 V et de son support avec le connecteur cylindrique. Pour tester ta boule de cristal, approche ta main en faisant des mouvements de vague au-dessus. Elle devrait s'illuminer uniquement quand tes mains sont proches de la feuille d'aluminium du support. Rappelle-toi que tu devras ajuster la valeur du seuil de déclenchement jusqu'à obtention de l'interaction désirée.

Plus d'aventures avec les bibliothèques

Tu viens d'apprendre à te lancer seul et à utiliser les bibliothèques de fonctions créées par d'autres personnes avec ton Arduino. Tout un monde de possibilités s'ouvre à toi ! Explore le contenu du site `http://playground.arduino.cc/` afin de découvrir les autres bibliothèques disponibles.

Si tu veux en savoir plus sur l'utilisation des capteurs tactiles capacitifs, visite le site suivant en langue anglaise : `http://playground.arduino.cc/CapacitiveSensor`.

Tu peux aussi rechercher en ligne des sites en français, avec les mots clés « capteurs capacitifs Arduino ». Il en existe beaucoup qui traitent de ce sujet ou qui commercialisent des capteurs.

Copyright © 2015 Eyrolles.

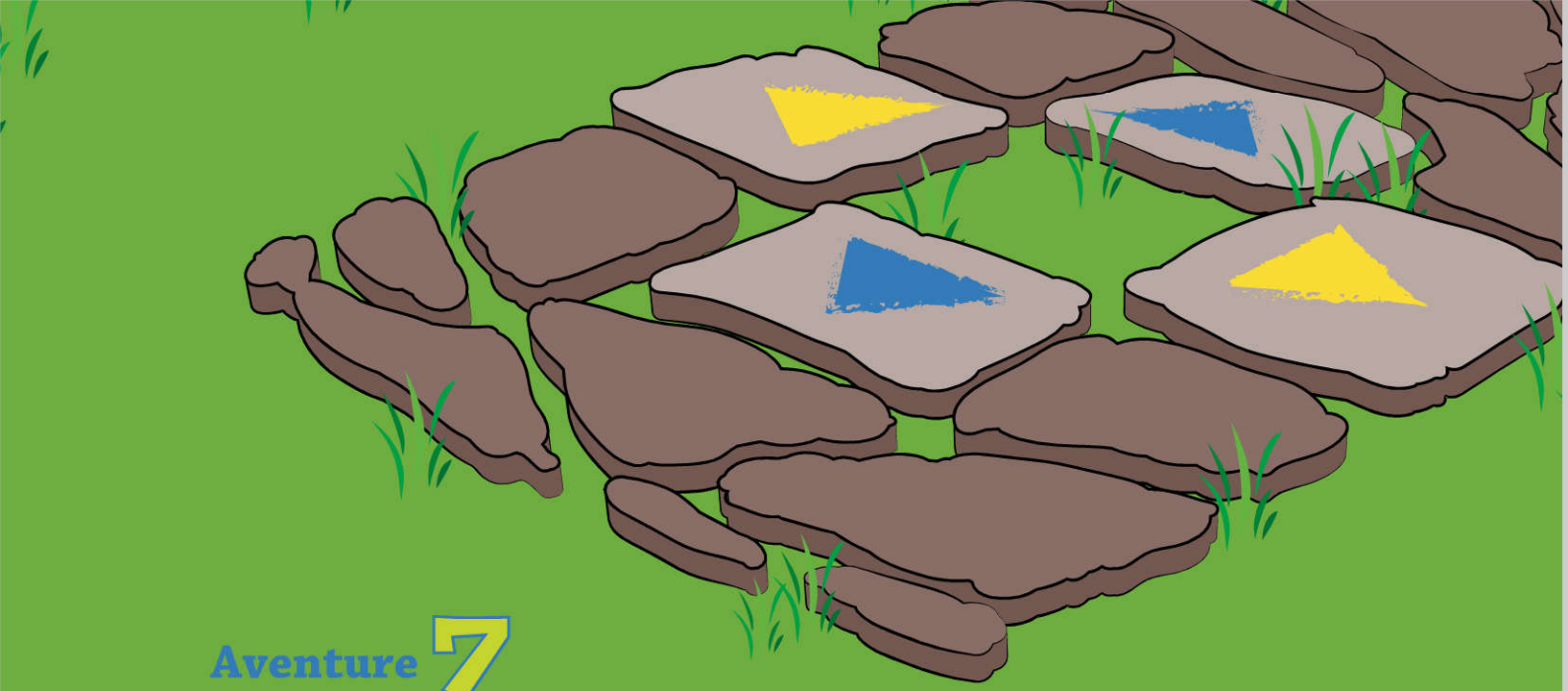
Rappel des commandes Arduino de cette aventure	
Commande	**Description**
`analogWrite()`	Envoie une tension comprise entre 0 et 5 V sur une broche déterminée. Une valeur de 0 provoque la sortie d'une tension de 0 V et une valeur de 255, une sortie de 5 V. Voir également `http://arduino.cc/en/Reference/AnalogWrite`.
`CapacitiveSensor`	Bibliothèque à utiliser avec des capteurs capacitifs. Voir également `http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense`.
`long`	Type de données pouvant mémoriser un nombre entier compris entre –2 147 483 648 et 2 147 483 647. Voir également `http://arduino.cc/en/Reference/Long`.
`map()`	Reçoit une valeur comprise dans des limites initiales et la transforme en une valeur contenue dans de nouvelles limites. Voir également `http://arduino.cc/en/Reference/Map`.

Étape franchie : Tu as étendu tes connaissances aux bibliothèques de code !

Dans la prochaine aventure

Tu découvriras d'autres types de cartes Arduino et tu créeras une manette de jeu personnalisée pour ton ordinateur !

Copyright © 2015 Eyrolles.



Aventure 7

Utilise l'Arduino Leonardo

EN CHOISISSANT la carte Arduino Uno dans le menu Outils > Type de carte de l'IDE Arduino, tu as probablement remarqué qu'il y a une longue liste d'autres cartes, qui paraissent toutes très intéressantes. Qu'est-ce qui les différencie de l'Arduino Uno ? Je n'ai pas assez de place dans ce livre pour te les décrire toutes, mais dans cette aventure, tu vas te familiariser avec une nouvelle carte : la Leonardo. Celle-ci a une caractéristique particulière que ne possède pas la Uno : elle peut apparaître sur ton ordinateur comme étant un clavier ou une souris USB.

Pour cette aventure, tu vas utiliser un capteur qui peut déterminer s'il est en pleine lumière ou à l'ombre, pour créer une manette USB de contrôle de jeu sur ton ordinateur. Ce capteur va te permettre de jouer en effectuant des mouvements de vagues avec tes mains au-dessus de ta manette !

Ce dont tu as besoin

- Un ordinateur ;
- une carte Arduino Leonardo ;
- un câble micro-USB ;
- une plaque de montage ;

Copyright © 2015 Eyrolles.

- 4 fils de pontage ;
- une photorésistance ;
- une résistance de 10 kΩ.

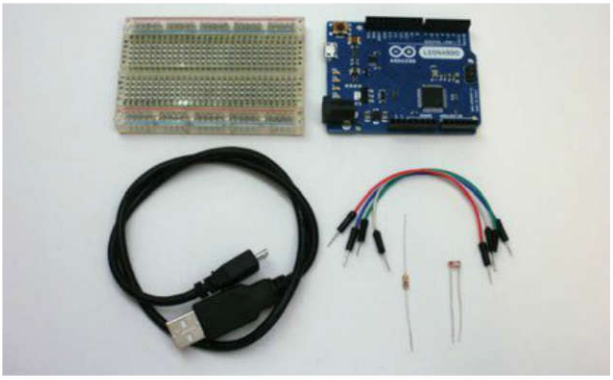


FIGURE 7-1 Les composants électroniques dont tu as besoin pour la première partie de cette aventure.

Découverte de l'Arduino Leonardo

Une des caractéristiques importantes de la plate-forme Arduino est la possibilité de choisir différentes cartes pour chaque projet, sans avoir à changer le code. Tu peux utiliser presque n'importe quel programme que tu as écrit pour la Uno avec une Leonardo. Tu utilises d'ailleurs le même environnement de développement (l'IDE Arduino) pour télécharger tes programmes.

La principale différence porte sur la sélection de la carte que tu utilises dans le menu Outils > Type de carte dans l'IDE Arduino. Tu dois maintenant choisir Arduino Leonardo, comme sur la figure 7-2.

Première connexion de ta carte Arduino Leonardo

La première particularité que tu vas noter en reliant ta carte Leonardo pour la première fois est qu'elle ne possède pas le même type de connecteur USB qu'une carte Arduino Uno. Tu ne peux donc pas utiliser le même câble pour les deux cartes.

Les connecteurs des câbles USB peuvent avoir différentes dimensions et formes, comme le montre la figure 7-3. Ce que tu peux considérer comme un câble USB « normal » possède un connecteur de type A à l'une de ses extrémités qui se branche sur ton ordinateur, et un connecteur de type B à l'autre extrémité que tu branches sur ta carte Arduino Uno. La carte Arduino Leonardo utilise un connecteur micro-USB au lieu du

Copyright © 2015 Eyrolles.

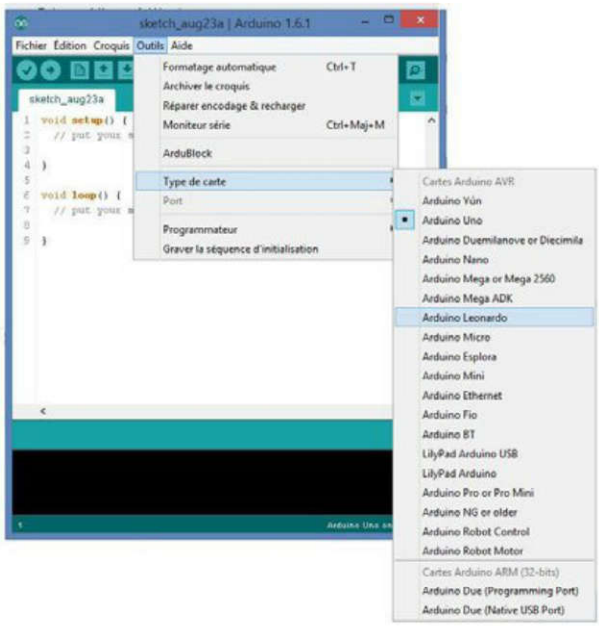


FIGURE 7-2 Sélection de la carte Arduino Leonardo depuis le menu Outils > Type de carte de l'IDE Arduino.

connecteur de type B. Ceci ne change rien par rapport à un câble USB normal : la carte Arduino Leonardo est aussi alimentée par l'ordinateur et peut communiquer avec lui. La forme du connecteur est sa seule différence. Tu dois maintenant utiliser deux câbles différents et les conserver !

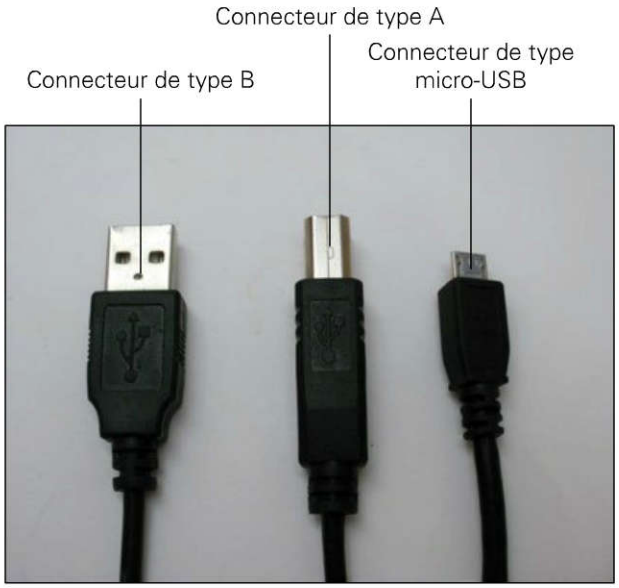


FIGURE 7-3 Les différents connecteurs USB.

Copyright © 2015 Eyrolles.

Une des caractéristiques que la carte Leonardo possède et que n'a pas la Uno lui permet d'être vue par l'ordinateur comme un clavier USB. Pour cette raison, quand tu la relies à ton ordinateur, tu peux voir apparaître une fenêtre te disant que celui-ci a détecté un nouveau clavier. Ferme-la simplement. Tu n'as pas à cliquer sur Continuer ou Installation d'un nouveau périphérique, car tu vas programmer ta carte en utilisant l'IDE Arduino, comme tu le faisais avec l'Arduino Uno.

Si l'ordinateur que tu utilises pour programmer ton Arduino Leonardo fonctionne sous Windows 7, 8 ou 10, tu devras peut-être installer des pilotes additionnels. Relie ta carte Leonardo et attends que l'installation des drivers se termine. Si rien ne se passe après quelques minutes, rends-toi sur le site `http://arduino.cc/en/Guide/ArduinoLeonardoMicro?from=Guide.ArduinoLeonardo`, où tu trouveras des instructions (en anglais) pour installer les drivers manuellement. Tu peux aussi consulter cet article en français : `http://prod.openclassrooms.com/forum/sujet/port-serie-impossible`.

EXPLORATION DU CODE

La plupart du temps, le code que tu as écrit pour ton Arduino Uno fonctionne de façon identique avec la Leonardo. Il est cependant utile de savoir ce qui pourrait différer.

Avec la Uno, la carte repart à zéro (Reset) chaque fois que tu ouvres le moniteur série de l'IDE Arduino. La fonction `setup()` est alors exécutée une fois puis la fonction `loop()`. Ça ne fonctionne pas ainsi pour la carte Leonardo : elle ne redémarre pas lorsque le moniteur série est ouvert. Cela signifie que si tu veux afficher des données créées dans la fonction `setup()`, elles n'apparaîtront pas après avoir ouvert le moniteur série. De plus, l'appel de l'instruction `print` présente dans `setup()` sera probablement déjà passé et tu ne verras que les messages issus de la fonction `loop()`.

Dans l'IDE Arduino, crée un nouveau sketch avec le code suivant. Télécharge-le dans ta carte Leonardo et ouvre le moniteur série :

```
void setup() {
  Serial.begin(9600);
  Serial.println("Salut depuis setup !");
}

void loop() {
  Serial.println("Et salut depuis loop !");
  delay(1000);
}
```

Copyright © 2015 Eyrolles.

Tu ne vois que le message « Et salut depuis loop ! » affiché de façon répétitive.

Appuie maintenant sur le bouton « Reset » (remise à zéro ou redémarrage) de ta carte Leonardo, il se trouve près du connecteur micro-USB. Le moniteur série cesse alors totalement d'afficher les messages issus de la fonction `loop()` ! Ceci est dû au fait que la connexion entre le moniteur série et la carte Leonardo a été interrompue lorsque tu as appuyé sur le bouton de redémarrage. Tu dois fermer puis rouvrir le moniteur série pour que ta carte sache que le moniteur série est présent et en attente.

Que conclure de tout cela ? Cela signifie-t-il que tu ne pourras jamais afficher dans le moniteur série des messages en provenance de la fonction `setup()` de l'Arduino Leonardo ? Ce serait très décevant ! C'est très utile de pouvoir afficher des messages, cela t'aide à savoir ce qui se passe dans ta carte pour corriger les problèmes de ton programme. Bien heureusement, tu vas pouvoir le contourner.

Télécharge le programme suivant dans ton Arduino Leonardo et ouvre le moniteur série :

```
void setup() {
  Serial.begin(9600);
  while(! Serial); /* Attente de la présence de
  la connexion série. */
  Serial.println("Salut depuis setup !");
}

void loop() {
  Serial.println("Et salut depuis loop !");
  delay(1000);
```

Tu devrais maintenant voir le message « Salut depuis setup ! » s'afficher avant les messages « Et salut depuis loop ! ». Si tu appuies sur le bouton Reset de ta carte, tu vois qu'il est à nouveau nécessaire de fermer puis de rouvrir le moniteur série pour voir, encore une fois, les messages s'afficher.

Un comportement identique à celui d'un clavier

Tu peux envoyer des messages à ton ordinateur depuis ta carte Arduino Uno en utilisant par exemple `Serial.println()`, mais tu as alors besoin d'un programme spécial comme le moniteur série de l'IDE Arduino pour pouvoir lire ces messages. La Leonardo peut envoyer des messages qui n'ont pas besoin d'un programme spécial pour être compris par l'ordinateur. Cette carte peut en effet envoyer des messages qui simulent l'appui sur les touches d'un clavier. Tout programme qui réagit à l'appui des touches d'un clavier (comme un traitement de texte ou un jeu) peut donc les interpréter.

Copyright © 2015 Eyrolles.

Commence par créer un nouveau sketch dans l'IDE Arduino. Entre ensuite le code suivant pour créer une fonction setup() et une fonction `loop()` vides :

```
void setup() {

}

void loop() {

}
```

Avant de faire fonctionner ton Arduino Leonardo comme un clavier, il est très important de l'arrêter. Ta carte Leonardo se comporte comme le clavier de ton ordinateur et prend le dessus, de la même façon que lorsque tu connectes un nouveau clavier USB à ton ordinateur. Si elle envoie constamment des messages clavier à ton ordinateur, il peut être difficile de lui dire de faire quoi que ce soit d'autre et même de télécharger un nouveau sketch dans ta carte.

Afin d'éviter que cela ne se produise, vérifie d'abord dans la fonction `loop()` si une broche d'entrée numérique est mise à 0. Si c'est le cas, la carte n'envoie aucun message clavier. Tu n'as même pas besoin de câbler un interrupteur comme dans l'aventure 3 : utilise simplement un fil de pontage ! Voici comment faire.

Au début de ton sketch, avant la fonction `setup()`, crée une nouvelle variable pour mémoriser le numéro de la broche d'interruption :

```
int brocheInter = 4 ;
```

Entre les accolades `{ }` de `setup()`, initialise cette broche en mode `INPUT` (entrée) avec résistance de maintien :

```
pinMode(brocheInter, INPUT_PULLUP) ;
```

Puis, dans la fonction `loop()`, ajoute le code suivant afin de vérifier si la broche `brocheInter` a été reliée à la masse GND. Si c'est le cas, ton programme continue en affichant un message ; sinon, il ne fait rien :

```
// lecture de l'interrupteur
int etatInter = digitalRead(brocheInter) ;
// si l'interrupteur est ouvert (non relié à la masse GND),
if (etatInter == LOW) {
  // ajouter le programme du clavier ici
}
delay(500) ;
```

L'instruction `delay()` placée à la fin est destinée à ralentir la répétition de la boucle. De cette façon, les messages ne sont pas envoyés trop rapidement.

Copyright © 2015 Eyrolles.

Si tu relies maintenant un fil de pontage entre la broche `brocheInter` (4) et la masse GND, le code que tu as inséré à l'intérieur de l'instruction `if` est exécuté. Si tu ne connectes rien à `brocheInter`, la résistance de maintien force la lecture d'un 1 sur `brocheInter` et le contenu de l'instruction `if` n'est pas exécuté.

Il est temps de commencer à ajouter des messages clavier !

La fonction `Keyboard` ressemble beaucoup à la fonction `Serial`. Dans `setup()`, tu dois initialiser l'envoi de messages clavier en appelant :

```
Keyboard.begin() ;
```

Ensuite, tu peux écrire tes messages avec la fonction `Keyboard` :

```
Keyboard.println ("Voici ta Leonardo se comportant comme ↩
  un clavier.") ;
```

En assemblant toutes ces instructions, tu obtiens le sketch suivant :

```
int brocheInter = 4 ;

void setup() {
  // initialise la broche en mode entrée et résistance de
  // maintien
  pinMode(brocheInter, INPUT_PULLUP) ;
  // initialisation du clavier
  Keyboard.begin();
}

void loop() {
  // lecture de l'interrupteur
  int etatInter = digitalRead(brocheInter);

  // si l'interupteur est ouvert (non relié à à la masse GND)
  if (etatInter == LOW) {
    Keyboard.println ("Voici ta Leonardo se comportant ↩
    comme un clavier.") ;
  }
delay(500) ;
}
```

Télécharge le sketch dans ta carte Arduino Leonardo puis démarre un programme de traitement de texte. Utilise le fil de pontage pour relier la broche `brocheInter` à la masse GND. Tu devrais alors voir ta Leonardo envoyer des messages clavier comme sur la figure 7-4 !

Copyright © 2015 Eyrolles.

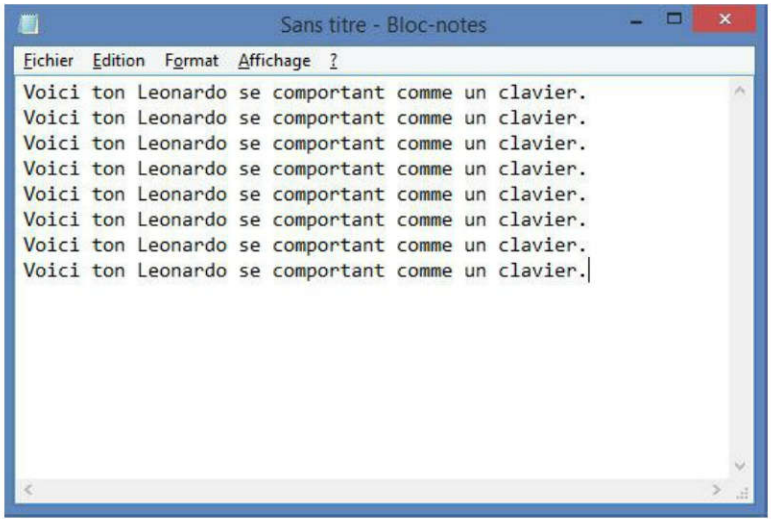


FIGURE 7-4 L'Arduino Leonardo envoyant des messages clavier dans un programme de traitement de texte.

Ne crée jamais de programme clavier qui ne pourra être interrompu ! Si c'est le cas et que tu rencontres des problèmes en téléchargeant un nouveau sketch dans ta carte Arduino Leonardo, tu peux essayer de remettre manuellement à zéro ta carte et lui demander de se mettre en attente de recevoir le sketch. Pour cela, maintiens son bouton Reset et clique sur le bouton Téléverser de l'IDE Arduino. Attends jusqu'à ce que l'IDE t'informe du téléversement puis lâche le bouton. Ton sketch devrait alors terminer de se télécharger dans ta carte.

Sais-tu que si tu as une idée nouvelle d'utilisation d'une carte Arduino, tu peux la développer et vendre le produit dans une boutique ? Makey Makey est un exemple de projet Arduino que tu peux acheter en magasin. C'est un produit qui fonctionne avec une carte Arduino Leonardo – son nom vient de « *Make anything into a keyboard* » (fabrique un clavier à partir de n'importe quoi). Tu connais déjà le code qui le fait fonctionner, mais si tu veux en savoir davantage, tu peux consulter le site (en anglais) `http://makeymakey.com` ou le site (en français) `http://blog.touchedeclavier.com/news/transformer-objet-touche-clavier/`. Tout ceci est possible parce qu'Arduino est un matériel libre et son logiciel également. Pour mieux comprendre ce que cela veut dire, consulte les sites (en français) `https://fr.wikipedia.org/wiki/Matériel_libre` et `https://fr.wikipedia.org/wiki/Open_source`.

Copyright © 2015 Eyrolles.

Détection de la lumière

Dans l'aventure 2, tu as étudié les entrées analogiques en utilisant un potentiomètre pour piloter un panneau d'information. Voici un petit rappel au sujet du fonctionnement de ce type d'entrée.

L'entrée analogique d'une carte Arduino est une broche qui peut lire des tensions comprises entre 0 V (la masse GND) et 5 V. Une entrée numérique peut lire soit la masse GND, soit 5 V. Elle ne peut pas savoir si l'entrée n'est qu'à 2,3 V ou à toute autre valeur comprise entre la masse GND et 5 V (voir la figure 7-5).

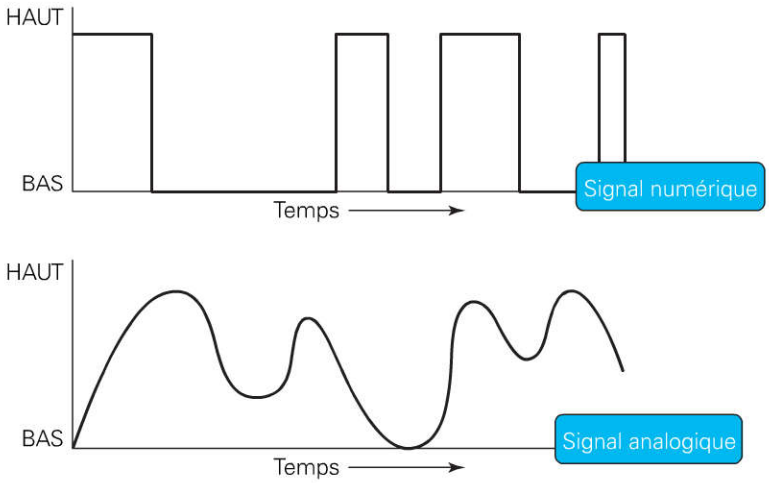


FIGURE 7-5 Signaux analogique et numérique.

Il existe toutes sortes de capteurs qui délivrent des sorties analogiques et non pas des signaux numériques. C'est-à-dire qu'ils ne savent pas uniquement si quelque chose est présent ou non : ils mesurent également l'intensité de ce qu'ils détectent. Un microphone mesure un son, un accéléromètre mesure un mouvement et un capteur de lumière mesure l'intensité lumineuse. Tous ces capteurs délivrent un signal analogique qui doit être lu par une entrée analogique de ta carte Arduino. Le capteur de luminosité que tu vas utiliser dans cette aventure est de type LDR *(Light Dependant Resistor)* ou **photorésistance**. Une photorésistance est une résistance qui change de valeur selon la quantité de lumière qu'elle reçoit. Je préfère le terme LDR (que l'on traduit par « résistance photo dépendante ») plutôt que photorésistance, car il décrit exactement son fonctionnement : sa dépendance à la luminosité.

Une **photorésistance** (**LDR**) est un composant dont la résistance varie selon l'intensité lumineuse à laquelle elle est exposée. On la nomme parfois résistance photodépendante.

Copyright © 2015 Eyrolles.

Utilise ton multimètre pour mesurer la résistance de ta photorésistance. Relie la pointe de touche de son cordon de mesure à un des fils de ta photorésistance et l'autre pointe du cordon de mesure à son autre fil. Règle le multimètre sur la mesure des résistances et regarde la valeur obtenue. Essaie d'exposer ta LDR à plus de lumière et vois ce qui se passe. Que se passe-t-il si tu recouvres ta photorésistance pour bloquer la lumière ?

Réalisation de ton circuit

Une carte Arduino peut uniquement mesurer des tensions et une LDR ne peut que modifier sa résistance, ce qui complique un peu les choses. Comment faire en sorte que ton Arduino puisse évaluer la façon dont ta LDR réagit à un environnement lumineux intense ou faible, en modifiant la valeur de sa résistance et non une tension ? La solution est d'utiliser un circuit très astucieux.

La tension, l'intensité et la résistance sont étroitement liées. Tu ne peux pas changer une de ces valeurs dans un circuit sans que les autres soient modifiées. La manière dont elles sont reliées est définie par la **loi d'Ohm** illustrée par la figure 7-6.

La loi d'Ohm

$$U = RI$$

Tension en volts = Intensité en ampères x Résistance en ohms

peut également s'exprimer ainsi :

$$R = \frac{U}{I}$$

$$\text{Résistance en ohms} = \frac{\text{Tension en volts}}{\text{Intensité en ampères}}$$

FIGURE 7-6 La Loi d'Ohm définit la relation entre tension, intensité et résistance.

La **loi d'Ohm** est la relation mathématique entre la tension, l'intensité et la résistance. La tension est égale à l'intensité multipliée par la résistance, ou, symboliquement, U=RI.

Tu n'as pas à te soucier davantage de cette équation pour le moment. Le plus important est de savoir comment tu peux définir un circuit qui va changer la tension quand la résistance est modifiée. Le type de circuit que tu vas réaliser s'appelle un **pont diviseur de tension**.

Copyright © 2015 Eyrolles.

Un **pont diviseur de tension** est un circuit qui génère une partie de la tension qui lui est appliquée. C'est un circuit très utile pour traduire le changement de valeur d'une résistance en changement de tension.

Le pont diviseur de tension que tu vas réaliser comporte deux résistances. L'une d'elles est une résistance « normale » de valeur fixe, l'autre est une résistance variable, dont la valeur change. Tu vas ainsi utiliser ta photorésistance comme résistance variable. Cependant, dans un projet futur, tu pourras utiliser ce circuit avec un autre capteur dont la résistance varie.

Quand la résistance de ta LDR augmente ou diminue, la tension de sortie augmente ou diminue également. Ainsi, quand plus ou moins de lumière éclaire ta photorésistance, la tension de sortie varie. L'endroit où tu places la photorésistance, en haut ou en bas du circuit, affecte la façon dont la tension varie quand tu bloques la lumière qui l'atteint. La figure 7-7 montre deux manières d'utiliser une LDR dans un diviseur de tension.

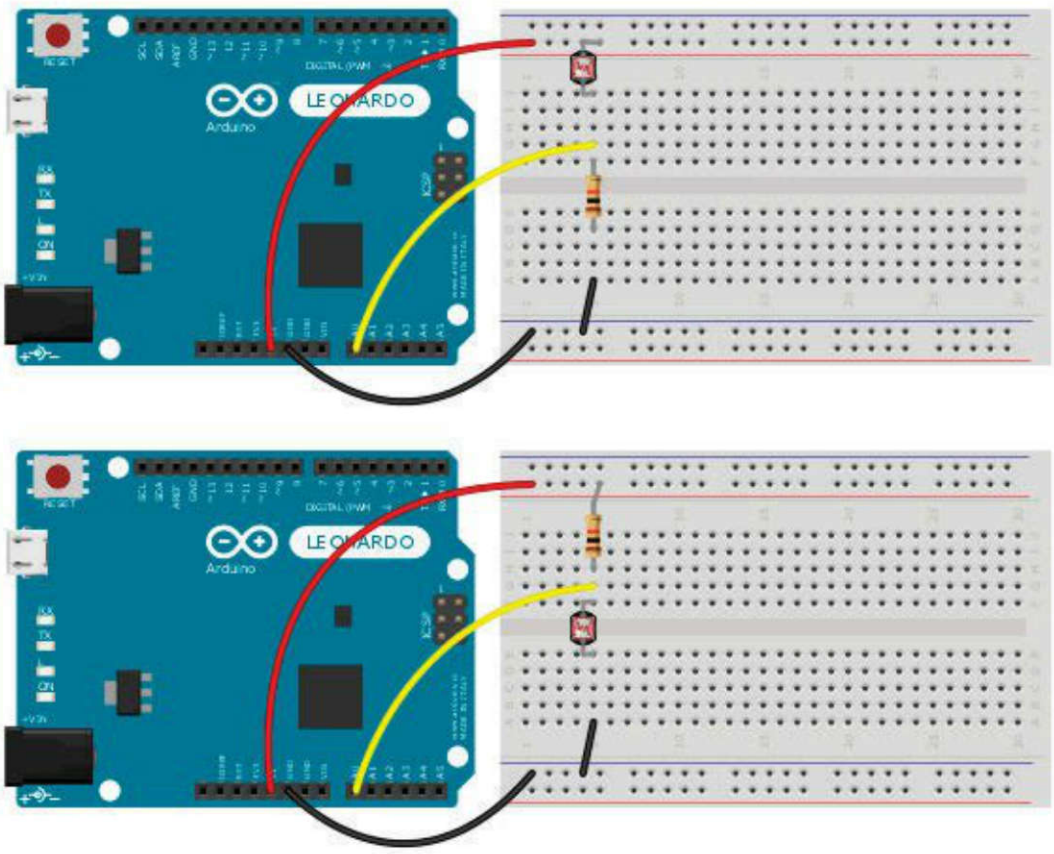
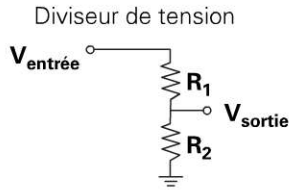


FIGURE 7-7 Les deux circuits diviseurs de tension, l'un avec la photorésistance en haut et l'autre avec la photorésistance en bas.

Copyright © 2015 Eyrolles.

Certaines personnes trouvent qu'il est plus facile de comprendre comment quelque chose fonctionne en le décrivant à l'aide d'équations mathématiques. Si tu veux calculer de combien la tension de sortie du circuit varie quand tu changes la valeur des résistances, tu peux utiliser les équations de la figure 7-8. Si tu n'aimes pas travailler avec des nombres et des équations, tu peux simplement ignorer tout cela pour l'instant et construire directement le circuit puis constater ce qui se passe.



$$V_{sortie} = V_{entrée} \times \frac{R_2}{R_1 + R_2}$$

$$\text{Tension de sortie en volts} = \text{tension d'entrée en volts} \times \frac{\text{résistance du bas en ohms}}{\text{résistance du haut en ohms}}$$

FIGURE 7-8 Les équations permettant de calculer comment différentes valeurs de résistance affectent la tension de sortie d'un diviseur de tension.

Écriture du programme

Une des caractéristiques importantes des cartes Arduino est leur indépendance vis-à-vis des capteurs qui y sont reliés. Ce qui compte, c'est uniquement de savoir s'il s'agit d'un capteur délivrant un signal analogique ou un signal numérique. Si tu as complété l'aventure 2, tu as déjà étudié le programme nécessaire à la lecture des valeurs retournées par ta photorésistance : c'est le même code que pour lire le signal provenant d'un potentiomètre. Ouvre l'exemple à partir du menu Fichier > Exemples > 01. Basics > AnalogRead de l'IDE Arduino et regarde ce sketch qui lit les valeurs sur l'entrée analogique `A0` et les affiche dans le moniteur série.

Dans le sketch `AnalogRead`, la fonction `setup()` initialise la communication série.

La version originale de l'exemple :

```
// the setup routine runs once when you press Reset:
void setup() {
  // initialize serial communication at 9600 bits per
  // second:
  Serial.begin(9600);
}
```

Copyright © 2015 Eyrolles.

La version francisée :

```
// setup s'exécute une fois après avoir appuyé sur le bouton
// Reset.
 void setup() {
  // initialisation de la communication série à 9600 bps :
  Serial.begin(9600);
}
```

La fonction `loop()` lit la valeur reçue sur la broche d'entrée analogique A0 et la mémorise dans une variable nommée `sensorValue` (ou `valeurCapteur`). Cette variable est ensuite affichée dans le moniteur série :

La version originale :

```
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1); // delay in between reads for stability
}
```

La version francisée :

```
// La function loop s'exécute indéfiniment :
void loop() {
  // Lit la valeur d'entrée de la broche analogique A0 :
  int valeurCapteur = analogRead(A0);
  // Affiche la valeur lue :
  Serial.println(valeurCapteur);
  delay(1); /* délai entre les lectures assurant plus de
  stabilité */
}
```

DÉFI

Câble le diviseur de tension en mettant la photorésistance en position haute et télécharge le sketch `AnalogSerial` dans ta carte Arduino. Ouvre le moniteur série et observe ce qui se passe quand tu illumines ou obscurcis le capteur.

Inverse la photorésistance et la résistance de valeur fixe afin que la LDR soit maintenant en position basse. Observe les valeurs affichées par le moniteur série. Quand penses-tu que le premier circuit devrait être utilisé et quand préférerais-tu utiliser le second ?

Copyright © 2015 Eyrolles.

Réalisation d'une manette de jeu

Tu sais maintenant comment faire en sorte qu'une carte Arduino Leonardo se comporte comme un clavier USB. Aimes-tu jouer aux jeux vidéo sur ton ordinateur ? Si c'est le cas, tu connais probablement des jeux auquels tu peux jouer avec ton clavier. Si tu n'y joues pas beaucoup, ce n'est pas un problème, je t'indiquerai quelques jeux que tu pourras tester.

Si tu sais rendre ton Arduino Leonardo semblable à un clavier contrôlant un jeu, alors tu peux commencer à créer ta propre manette comme sur la figure 7-9. Cela permet d'utiliser beaucoup de capteurs pour jouer. Fini, les boutons ennuyeux du passé ! Tu vas pouvoir réaliser une manette qui utilise la lumière pour contrôler l'envoi de caractères, mais tu pourrais aussi utiliser tout autre capteur de ta préférence !

Tu peux voir une vidéo (en anglais) montrant comment construire ta manette sur le site d'accompagnement (www.editions-eyrolles.com/go/arduino).

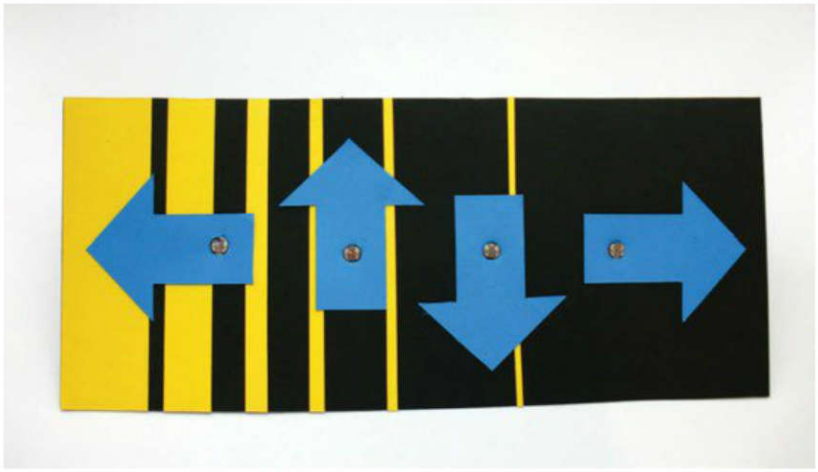


FIGURE 7-9 Une manette de contrôle de jeu utilisant un microcontrôleur Arduino Leonardo.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

Tu as besoin des fournitures suivantes pour réaliser ton contrôleur de jeu (la figure 7-10 montre les composants électroniques nécessaires) :

- un ordinateur ;
- une carte Arduino Leonardo ;
- un câble micro USB ;
- une plaque de montage ;
- 10 fils de pontage ;
- 4 résistances de 10 kΩ ;
- 4 photorésistances ;
- un morceau de papier ou une carte pour recouvrir la plaque de montage ;
- des crayons-feutres ou des crayons de couleur ;
- une paire de ciseaux ou un couteau de bricolage.

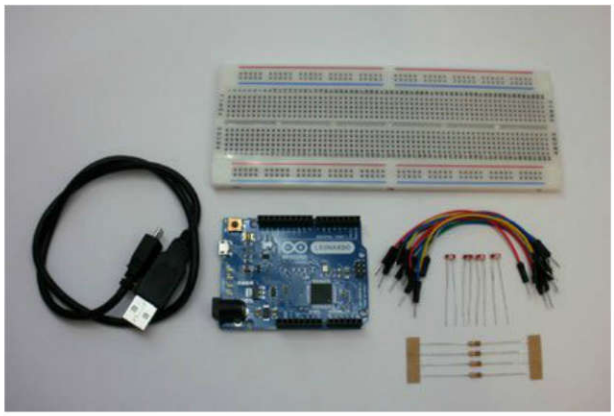


FIGURE 7-10 Les composants électroniques nécessaires à la réalisation de ta manette de jeu.

Construction de ton circuit

Tu n'as pas besoin de souder des fils pour réaliser le circuit de ta manette de contrôle, tu peux le câbler directement sur ta plaque de montage.

Le circuit complet de ta manette est composé d'un même circuit répété quatre fois. Il contient quatre photorésistances, chacune contrôlant une flèche directionnelle et chacune intégrée à son propre diviseur de tension, qu'elles partagent avec une résistance de valeur fixe. Le schéma du circuit de la manette de jeu est représenté sur la figure 7-11.

Copyright © 2015 Eyrolles.

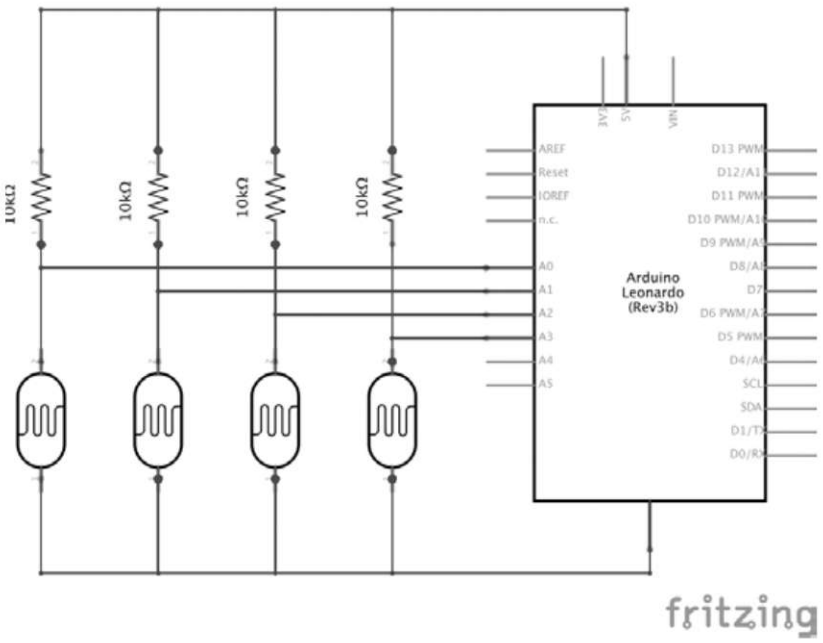


FIGURE 7-11 Le schéma du circuit de ta manette de contrôle de jeu.

Suis les étapes ci-dessous pour réaliser le circuit de la figure 7-12 sur ta plaque de montage. Je te recommande d'espacer les quatre photorésistances, car elles ne doivent pas être trop proches les unes des autres. En effet, il n'est pas souhaitable que tu puisses bloquer accidentellement la lumière d'une photorésistance alors que tu souhaites bloquer l'autre.

1. Positionne les quatre photorésistances à cheval sur l'espace séparateur de ta plaque de montage, en les espaçant régulièrement sur toute la longueur de la plaque.
2. Relie chacune des résistances entre une des grandes rangées situées en haut de ta plaque de montage et les photorésistances.
3. Connecte un fil de pontage entre le fil du bas de chaque photorésistance et une grande rangée située en bas de ta plaque.
4. Relie la grande rangée située en haut, dans laquelle les résistances sont insérées, à la broche 5 V de ta carte Arduino Leonardo.
5. Utilise un fil de pontage pour relier la grande rangée située en bas, dans laquelle les fils de pontage provenant des photorésistances sont branchés, à la broche de masse GND de ta carte Leonardo.
6. Utilise quatre fils de pontage pour relier les broches A0, A1, A2 et A3 à chacune des photorésistances. Une des extrémités de chaque fil de pontage doit être insérée dans la broche de la Leonardo et l'autre dans la rangée où la photorésistance et la résistance sont reliées.

Copyright © 2015 Eyrolles.

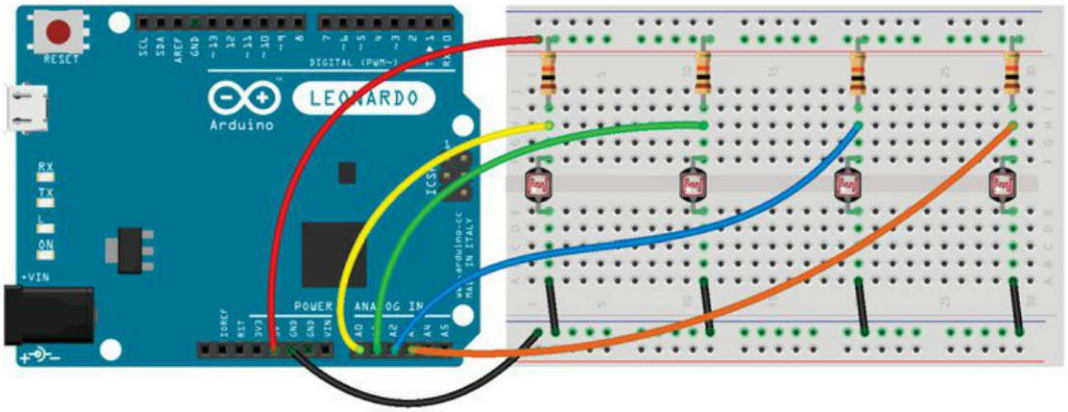


FIGURE 7-12 Réalisation du circuit de la manette de jeu.

Écriture du programme

Pour t'assurer que ton circuit fonctionne correctement, tu vas d'abord programmer ton Arduino Leonardo afin qu'elle affiche des messages sur le moniteur série, puis pour qu'elle se comporte comme un clavier.

Commence par créer un sketch vierge comportant les fonctions `setup()` et `loop()` vides.

```
void setup() {
}

void loop() {
}
```

Au début du sketch, avant la fonction `setup()`, déclare et initialise les variables indiquant les broches d'entrée. Il y en a cinq : la broche de l'interrupteur et une entrée pour chaque photorésistance.

La version originale :

```
int switchPin = 4;
int leftSensor = A0;
int rightSensor = A1;
int upSensor = A2;
int downSensor = A3;
```

La version francisée :

```
int brocheInter = 4;
int capteurGauche = A0;
int capteurDroit = A1;
```

Copyright © 2015 Eyrolles.

```
int capteurHaut = A2;
int capteurBas = A3;
```

Dans la fonction `setup()`, démarre la communication série et prépare la broche `brocheInter` en mode `INUT_PULLUP` :

La version originale :

```
// make the switchPin an input with an internal pull-up
// resistor
pinMode(switchPin, INPUT_PULLUP);
// initialize control over the keyboard:
Serial.begin(9600);
```

La version francisée :

```
// Mettre brocheInter en mode entrée avec résistance de
// maintien
pinMode(brocheInter, INPUT_PULLUP);
// Initialiser la communication série
Serial.begin(9600);
```

Dans la fonction `loop()`, vérifie l'état de la broche de l'interrupteur et crée une instruction if qui est exécutée si `brocheInter` est au niveau bas (`LOW`) :

La version originale :

```
// read the pushbutton:
int switchState = digitalRead(switchPin);
// if the switch is open (not connected to ground),
if (switchState == LOW) {

}
delay(50);
```

La version francisée :

```
// Vérifier la connexion de contrôle
int etatInter = digitalRead(brocheInter);
// Si le fil est relié à la masse GND,
if (etatInter == LOW) {

}
delay(50);
```

Le délai provoqué par `delay()` permet au message de ne pas être envoyé trop rapidement.

Au début de ton sketch, après les autres déclarations de variables, ajoute les lignes de code suivantes.

Copyright © 2015 Eyrolles.

La version originale :

```
int rightThreshold = 400;
int leftThreshold = 400;
int upThreshold = 400;
int downThreshold = 400
```

La version francisée :

```
int seuilDroit = 400;
int seuilGauche = 400;
int seuilHaut = 400;
int seuilBas = 400;
```

Ces variables mémorisent le seuil à partir duquel chaque photorésistance déclenchera l'envoi d'un message.

À l'intérieur de l'instruction `if` que tu as créée, ajoute le code suivant. Même si cela paraît long, c'est quatre fois le même bloc d'instructions, un pour chaque photorésistance.

Tu devras probablement utiliser une valeur différente de 400 pour tes photorésistances. Tu pourras augmenter ou diminuer cette valeur, chaque photorésistance pouvant avoir besoin d'une différente ! Tu détermineras plus tard, par essais successifs, les meilleures valeurs à adopter pour ta manette, mais tu n'en es pas encore là : tu dois d'abord terminer ton sketch.

La version originale :

```
// RIGHT ARROW
int rightValue = analogRead(rightSensor);
//Serial.println(rightValue);
if(rightValue > rightThreshold) {
  Serial.println("right arrow");
}

// LEFT ARROW
int leftValue = analogRead(leftSensor);
//Serial.println(leftValue);
if(leftValue > leftThreshold) {
  Serial.println("left arrow");
}
```

Copyright © 2015 Eyrolles.

```
// UP ARROW
int upValue = analogRead(upSensor);
//Serial.println(upValue);
if(upValue > upThreshold) {
  Serial.println("up arrow");
}

// DOWN ARROW
int downValue = analogRead(downSensor);
//Serial.println(downValue);
if(downValue > downThreshold) {
  Serial.println("down arrow");
}
```

La version francisée :

```
// Flèche droite
int valeurDroite = analogRead(capteurDroit);
//Serial.println(valeurDroite);
if(valeurDroite > seuilDroit) {
  Serial.println("flèche droite");
}

// Flèche gauche
int valeurGauche = analogRead(capteurGauche);
//Serial.println(valeurGauche);
if(valeurGauche > seuilGauche) {
  Serial.println("flèche gauche");
}

// Flèche Haut
int valeurHaut = analogRead(capteurHaut);
//Serial.println(valeurHaut);
if(valeurHaut > seuilHaut) {
  Serial.println("flèche haute");
}

// Flèche Bas
int valeurBas = analogRead(capteurBas);
//Serial.println(valeurBas);
if(valeurBas > seuilBas) {
  Serial.println("flèche basse");
}
```

Dans chacun des quatre blocs de code, la valeur de la broche d'entrée analogique est lue et mémorisée dans une variable. Si cette variable est supérieure au seuil de déclenchement de la photorésistance, un message est affiché.

Copyright © 2015 Eyrolles.

Dans chaque bloc de code, il y a une ligne de commentaire. Si tu supprimes la marque de commentaire (en effaçant les doubles barres de fraction //), la valeur de la broche sera affichée. Cela est utile pour t'aider à déterminer les valeurs de seuil, mais tout un ensemble de valeurs sera aussi imprimé et pourra te sembler confus. Tu préféreras sans doute n'afficher que les valeurs d'une seule photorésistance à la fois.

Tu trouveras ci-dessous le sketch complet te permettant d'afficher les messages. Construis ton circuit et télécharge ce programme dans ta carte Leonardo.

> Rappelle-toi que ton sketch ne lira les valeurs des broches analogiques que si tu relies la broche 4 à la masse GND.
>
>
>

La version originale :

```
int switchPin = 4;
int leftSensor = A0;
int rightSensor = A1;
int upSensor = A2;
int downSensor = A3;

// adjust these variables to values that
// work for your controller
int rightThreshold = 400;
int leftThreshold = 400;
int upThreshold = 400;
int downThreshold = 400;

void setup() {
  // make the switchPin an input
  // with an internal pull-up resistor
  pinMode(switchPin, INPUT_PULLUP);
  // initialize control over the keyboard:
  Serial.begin(9600);
}

void loop() {
  // read the pushbutton:
  int switchState = digitalRead(switchPin);

  // if the switch is open (not connected to ground),
  if (switchState == LOW) {
    // RIGHT ARROW
```

Copyright © 2015 Eyrolles.

```
    int rightValue = analogRead(rightSensor);
    //Serial.println(rightValue);
    if(rightValue > rightThreshold) {
    Serial.println("right arrow");
    }

    // LEFT ARROW
    int leftValue = analogRead(leftSensor);
    //Serial.println(leftValue);
    if(leftValue > leftThreshold) {
      Serial.println("left arrow");
    }

    // UP ARROW
    int upValue = analogRead(upSensor);
    //Serial.println(upValue);
    if(upValue > upThreshold) {
      Serial.println("up arrow");
    }

    // DOWN ARROW
    int downValue = analogRead(downSensor);
    //Serial.println(downValue);
    if(downValue > downThreshold) {
      Serial.println("down arrow");
    }
  }
  delay(50);
}
```

La version francisée :

```
int brocheInter = 4;
int capteurGauche = A0;
int capteurDroit = A1;
int capteurHaut = A2;
int capteurBas = A3;

// Ajuster la valeur de ces variables afin
// d'obtenir un fonctionnement correct de votre manette
int seuilDroit = 400;
int seuilGauche = 400;
int seuilHaut = 400;
int seuilBas = 400;

void setup() {
  // Mettre brocheInter en mode entrée avec résistance de
  // maintien
```

Copyright © 2015 Eyrolles.

```
  pinMode(brocheInter, INPUT_PULLUP);
  // Initialiser la communication série
  Serial.begin(9600);
}

void loop() {
  // Vérifier la connexion de contrôle
  int etatInter = digitalRead(brocheInter);

  // Si le fil est relié à GND,
  if (etatInter == LOW) {
    // Flèche droite
    int valeurDroite = analogRead(capteurDroit);
    //Serial.println(valeurDroite);
    if(valeurDroite > seuilDroit) {
      Serial.println("flèche droite");
    }

    // Flèche gauche
    int valeurGauche = analogRead(capteurGauche);
    //Serial.println(valeurGauche);
    if(valeurGauche > seuilGauche) {
      Serial.println("flèche gauche");
    }

    // Flèche Haut
    int valeurHaut = analogRead(capteurHaut);
    //Serial.println(valeurHaut);
    if(valeurHaut > seuilHaut) {
      Serial.println("flèche haute");
    }

    // Flèche Bas
    int valeurBas = analogRead(capteurBas);
    //Serial.println(valeurBas);
    if(valeurBas > seuilBas) {
      Serial.println("flèche basse");
    }
  }
  delay(50) ;
}
```

Comme ce programme est long, tu préféreras peut-être le télécharger (en anglais ou en français) depuis le site d'accompagnement (www.editions-eyrolles.com/go/arduino), plutôt que de le saisir au clavier.

Copyright © 2015 Eyrolles.

Ouvre maintenant le moniteur série et essaie tes photorésistances une par une, en maintenant ta main en face de chacune. Ajuste alors les valeurs de seuil afin de n'afficher un message que lorsque tu le veux.

Après avoir déterminé les valeurs des seuils, tu peux modifier ton sketch pour qu'il envoie des caractères du clavier à la place des messages.

Sauvegarde ton sketch puis crées-en un vide. Copie le contenu du sketch que tu viens de sauvegarder dans le nouveau sketch. Tu vas conserver la majeure partie de ce que tu as déjà écrit, mais tu dois remplacer les lignes de code qui utilisent la fonction `Serial` par les fonctions `Keyboard`.

Dans la fonction `setup()`, remplace la ligne qui initialise la communication série par le code suivant :

La version originale :

```
// initialize control over the keyboard
Keyboard.begin();
```

La version francisée :

```
// Initialisation des transmissions vers le clavier
Keyboard.begin();
```

Dans chacun des quatre blocs de code qui affichent les messages informant du déclenchement des capteurs, remplace l'instruction `Serial.println()` par le code suivant en utilisant `KEY_RIGHT_ARROW`, `KEY_LEFT_ARROW`, `KEY_UP_ARROW` et `KEY_DOWN_ARROW` :

```
Keyboard.press(KEY_RIGHT_ARROW);
```

Au lieu d'afficher un message, ta carte Arduino Leonardo va envoyer un message informant qu'une touche du clavier a été enfoncée. À la fin de `loop()`, la ligne de code suivante envoie un message disant que toutes les touches ont été relâchées :

```
Keyboard.releaseAll()
```

Après avoir ainsi modifié ton sketch, tu devrais obtenir le sketch suivant :

La version originale :

```
int switchPin = 4;
int leftSensor = A0;
int rightSensor = A1;
int upSensor = A2;
int downSensor = A3;
```

Copyright © 2015 Eyrolles.

```
// adjust these variables to values that
// work for your controller
int rightThreshold = 400;
int leftThreshold = 400;
int upThreshold = 400;
int downThreshold = 400;

void setup() {
  // make the switchPin and input
  // with an internal pull-up resistor
  pinMode(switchPin, INPUT_PULLUP);
  // initialize control over the keyboard:
  Keyboard.begin();
}

void loop() {
  // read the pushbutton:
  int switchState = digitalRead(switchPin);

  // if the switch is open (not connected to ground),
  if (switchState == LOW) {
    // RIGHT ARROW
    int rightValue = analogRead(rightSensor);
    //Serial.println(rightValue);
    if(rightValue > rightThreshold) {
      Keyboard.press(KEY_RIGHT_ARROW);
    }

    // LEFT ARROW
    int leftValue = analogRead(leftSensor);
    //Serial.println(leftValue);
    if(leftValue > leftThreshold) {
      Keyboard.press(KEY_LEFT_ARROW);
    }

    // UP ARROW
    int upValue = analogRead(upSensor);
    //Serial.println(upValue);
    if(upValue > upThreshold) {
      Keyboard.press(KEY_UP_ARROW);
    }

    // DOWN ARROW
    int downValue = analogRead(downSensor);
    //Serial.println(downValue);
    if(downValue > downThreshold) {
      Keyboard.press(KEY_DOWN_ARROW);
    }
```

Copyright © 2015 Eyrolles.

```
  }
delay(50);
Keyboard.releaseAll();
}
```

La version francisée :

```
int brocheInter = 4;
int capteurGauche = A0;
int capteurDroit = A1;
int capteurHaut = A2;
int capteurBas = A3;

// Ajuster la valeur de ces variables afin
// d'obtenir un fonctionnement correct de votre manette
int seuilDroit = 400;
int seuilGauche = 400;
int seuilHaut = 400;
int seuilBas = 400;

void setup() {
  // Mettre brocheInter en mode entrée avec résistance de
  // maintien
  pinMode(brocheInter, INPUT_PULLUP);
  // Initialiser la communication clavier :
  Keyboard.begin();
}

void loop() {
  // Vérifier la connexion de contrôle :
  int etatInter = digitalRead(brocheInter);

  // Si le fil est relié à GND,
  if (etatInter == LOW) {
    // Flèche droite
    int valeurDroite = analogRead(capteurDroit);
    //Serial.println(valeurDroite);
    if(valeurDroite > seuilDroit) {
      Keyboard.press(KEY_RIGHT_ARROW);
    }

    // Flèche gauche
    int valeurGauche = analogRead(capteurGauche);
    //Serial.println(valeurGauche);
    if(valeurGauche > seuilGauche) {
      Keyboard.press(KEY_LEFT_ARROW);
    }
```

Copyright © 2015 Eyrolles.

```
    // Flèche Haut
    int valeurHaut = analogRead(capteurHaut);
    //Serial.println(valeurHaut);
    if(valeurHaut > seuilHaut) {
      Keyboard.press(KEY_UP_ARROW);
    }

    // Flèche Bas
    int valeurBas = analogRead(capteurBas);
    //Serial.println(valeurBas);
    if(valeurBas > seuilBas) {
      Keyboard.press(KEY_DOWN_ARROW);     }
  }
  delay(50);
  Keyboard.releaseAll();
}
```

Réalise le circuit et télécharge le sketch. Ouvre un tableur, par exemple Microsoft Excel, et vérifie que tu peux utiliser ta manette de jeu pour déplacer ton curseur entre les différentes cases, comme tu le ferais avec les touches fléchées de ton clavier.

Ton jeu utilise-t-il d'autres touches que les flèches ? Par exemple, la barre d'espace ? Tu en sauras plus sur les différentes touches que ton Arduino Leonardo peut simuler en allant sur le site `http://arduino.cc/en/Reference/KeyboardModifiers`. Tu peux aussi consulter le site `http://forum.snootlab.com/viewtopic.php?f=32&t=605` qui explique, en français, l'utilisation des fonctions `Keyboard`.

Fabrication du boîtier de ta manette de jeu

Découpe la forme de ta manette en utilisant du papier et du carton. Ce peut être un rectangle ou toute autre forme. Il n'est pas nécessaire que cela ressemble aux manettes de jeu que tu as déjà vues. La seule contrainte est de lui permettre de cacher ta plaque de montage.

Décore ton boîtier comme tu le souhaites. Il serait judicieux que tu précises quelle touche correspond à chaque photorésistance par un marquage significatif. Il ne faudrait pas, en effet, que tu oublies quelle photorésistance masquer quand tu es à une phase critique de ton jeu, au risque de perdre la partie !

En utilisant des ciseaux, perce un trou aux endroits ou les photorésistances se trouveront (voir la figure 7-13). C'est dans ces trous que passeront les fils de chaque

Copyright © 2015 Eyrolles.

photorésistance qui sera placée au-dessus du couvercle, les fils étant reliés à ta plaque de montage sous le couvercle.

Assemble le tout

Puisque tu as déjà testé ta manette à l'aide d'un tableur, la dernière étape consiste à ajouter le coffret avant de jouer à ton jeu avec ta nouvelle manette.

Retire les photorésistances de ta plaque de montage une par une et place leurs fils au travers des trous percés dans le couvercle. Relie ensuite chaque photorésistance au circuit de la plaque de montage. Tu peux, si nécessaire, souder des fils plus longs aux photorésistances s'il est trop difficile de les relier à ta plaque de montage.

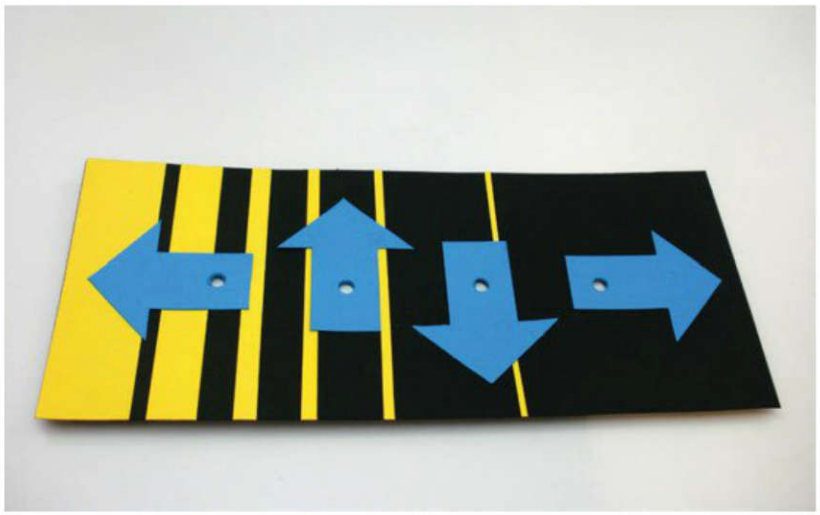


FIGURE 7-13 Le couvercle sans aucun circuit.

Teste ta manette à nouveau avec le tableur afin d'être sûr que tu as connecté le tout correctement. Après avoir vérifié que tout est correct, exécute ton jeu préféré utilisant les touches fléchées et commence à jouer !

Si tu cherches des jeux auxquels tu peux jouer en utilisant ta nouvelle manette, le jeu 2048, que tu peux trouver à l'adresse `http://gabrielecirulli.github.io/2048`, vaut le coup. C'est un jeu de réflexion avec des nombres. Tu peux aussi le trouver à l'adresse `http://2048-desktop.fr.softonic.com/telecharger`. J'aime également *A Maze Race*, qui se trouve à l'adresse `http://www.primarygames.com/puzzles/mazes/mazerace`. Beaucoup plus difficile qu'il n'y paraît !

Copyright © 2015 Eyrolles.

Plus d'aventures avec l'Arduino Leonardo

Tu viens d'apprendre à utiliser des circuits plus complexes appelés diviseurs de tension. Si tu veux en savoir davantage sur ce type de circuits, tu peux consulter le tutoriel de SparkFun (en anglais), à l'adresse `https://learn.sparkfun.com/tutorials/voltage-dividers`.

Pour obtenir plus d'informations sur la carte Arduino Leonardo et sur ce que tu peux en faire, tu peux visiter le site `http://arduino.cc/en/Guide/ArduinoLeonardoMicro?from=Guide.ArduinoLeonardo`.

Tu peux aussi voir d'autres d'exemples utilisant les fonctions `Keyboard` dans l'IDE Arduino dans Menu > Fichier > Exemples > 09.USB. Tu y trouveras également des exemples pour utiliser ta carte Arduino Leonardo comme une souris.

Les cartes Arduino Uno et Arduino Leonardo sont des matériels libres (open source). Cela signifie que tu peux réaliser tes propres projets et même de nouvelles cartes utilisant les fonctionnalités des microcontrôleurs Arduino, comme le Makey Makey. Si tu veux en savoir davantage sur les matériels libres, va sur le site de l'Open Source Hardware Association (OSHWA) à l'adresse `www.oshwa.org`.

Tu peux également trouver de nombreuses informations en français sur ce sujet sur Internet en utilisant un moteur de recherche avec les mots clés « matériel open source » ou « matériel libre ».

Rappel des commandes Arduino de cette aventure

Commande	Description
`While(!Serial);`	Provoque un arrêt du microcontrôleur Arduino Leonardo en attente de l'ouverture d'une communication série.
`Keyboard.begin()`	Démarrage des fonctionnalités clavier. Voir également `http://arduino.cc/en/Reference/KeyboardBegin`.
`Keyboard.println()`	Envoie un message à l'ordinateur comme s'il était entré au clavier. Voir également `http://arduino.cc/en/Reference/KeyboardPrintln`.
`Keyboard.press()`	Envoie un message à l'ordinateur indiquant qu'une touche clavier est enfoncée. Voir également `http://arduino.cc/en/Reference/KeyboardPress`.
`Keyboard.releaseAll()`	Envoie un message à l'ordinateur indiquant qu'aucune touche du clavier n'est enfoncée. Voir également `http://arduino.cc/en/Reference/KeyboardReleaseAll`.

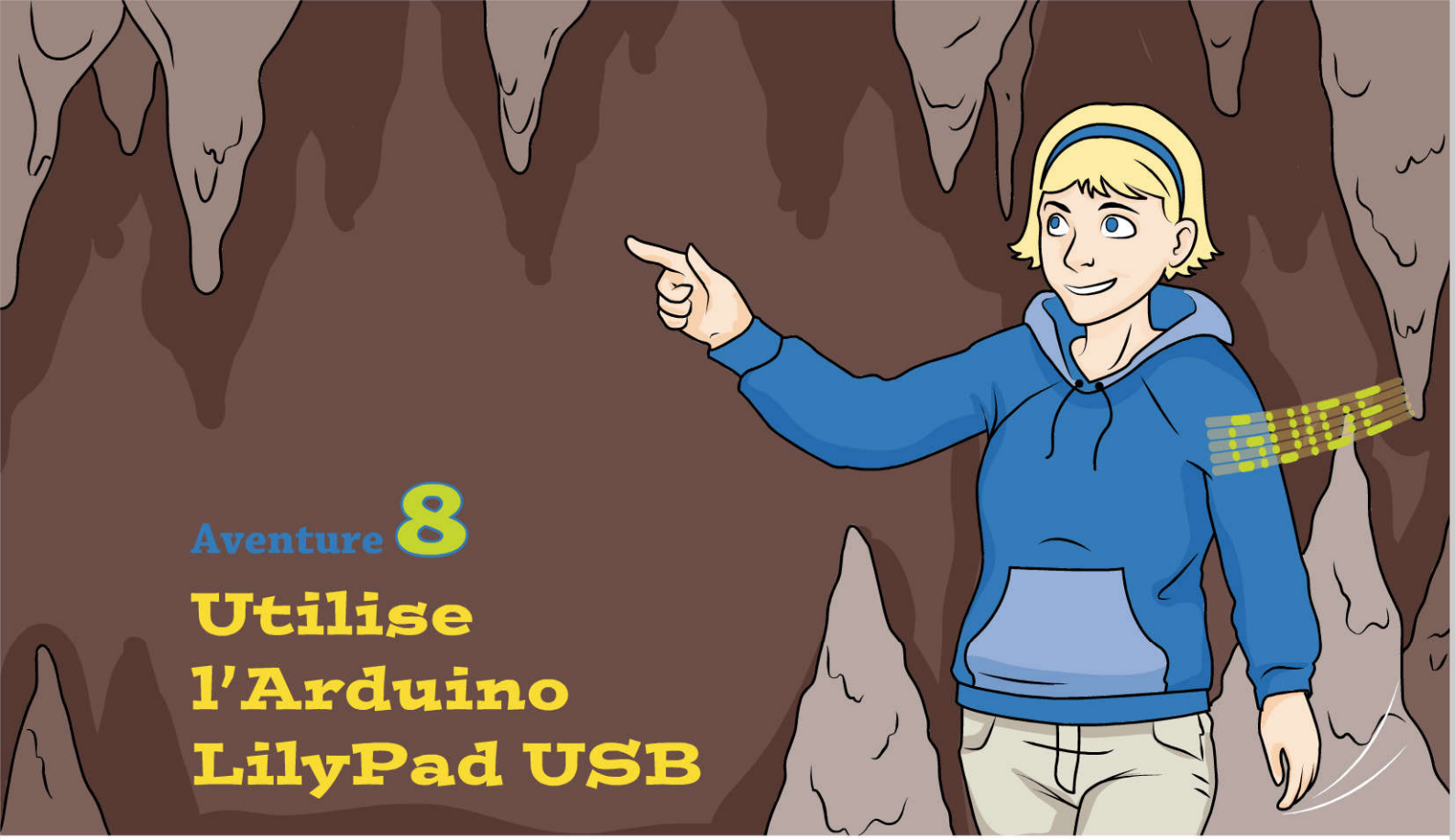
Copyright © 2015 Eyrolles.

Étape franchie : Tu es devenu un expert en réalisation de manettes de jeu. Bien joué !

Dans la prochaine aventure

Tu continueras à découvrir d'autres cartes Arduino en montant un circuit sur un vêtement que tu pourras porter !

Copyright © 2015 Eyrolles.



Aventure 8
Utilise l'Arduino LilyPad USB

TU VIENS À PEINE DE COMMENCER à explorer les différentes possibilités te permettant de réaliser un projet avec les microcontrôleurs Arduino ! Dans l'aventure 7, tu as appris à concevoir et à construire ta propre manette de jeu en utilisant une carte Arduino Leonardo. La Leonardo ressemble à la Uno. Tu as construis ton circuit de façon habituelle, en utilisant une plaque de montage, des fils et de la soudure.

Mais comment ferais-tu pour monter un circuit que tu devrais porter sur toi ? Tu ne peux pas employer de fil rigide ni de la soudure si tu veux quelque chose qui se plie comme un tissu et qui soit confortable à porter. Par chance, le métal conducteur peut être tressé, c'est-à-dire que tu peux coudre un circuit avec une aiguille et un fil à coudre conducteur. C'est ce que l'on appelle réaliser un circuit souple. De plus, il existe un type particulier de carte Arduino qui a été conçu pour faciliter l'intégration d'un microcontrôleur à ton projet : c'est l'Arduino LilyPad USB !

Dans cette aventure, tu vas créer un vêtement en incorporant une carte Arduino et des LED dans une manche. Les LED afficheront un message secret que tu programmeras dans ta carte Arduino LilyPad USB. Ce message y sera mémorisé sous la forme d'un tableau. Aussi, avant que tu n'écrives le code du programme de ce projet, je vais te rappeler le fonctionnement des tableaux et te montrer comment les étendre à la deuxième dimension.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

- Un ordinateur ;
- une carte Arduino LilyPad USB ;
- un câble micro-USB ;
- 10 câbles avec pinces crocodiles ;
- 5 LED LilyPad (ou 5 LED et 5 résistances de 68 ou 100 Ω).

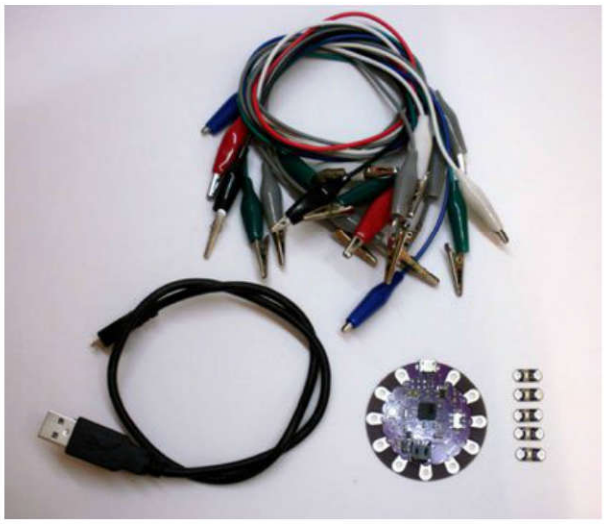


FIGURE 8-1 Les composants électroniques dont tu as besoin pour la première partie de cette aventure.

Découverte de l'Arduino LilyPad USB

L'Arduino LilyPad USB est un membre de la famille Arduino LilyPad. Bien que tu puisses utiliser n'importe laquelle des Arduino LilyPad, la LilyPad USB que tu vois sur la figure 8-2 possède quelques caractéristiques qui la rendent plus intéressante, telles que le fait de ne pas nécessiter une seconde carte pour télécharger les sketches. Comme ce sont toutes des Arduino, le code demeure identique : il fonctionnera indifféremment sur la LilyPad Simple ou sur la LilyPad Simple Snap.

L'Arduino LilyPad USB fonctionne avec le même microcontrôleur que la Leonardo. Cela signifie que tu peux l'utiliser pour un circuit simulant un clavier ou une souris comme tu pouvais le faire avec la Leonardo. Cela signifie également qu'elle est plus facile à programmer que les autres cartes Arduino LilyPad. La LilyPad USB possède un connecteur micro-USB, le même que celui de la Leonardo. Ainsi, programmer l'Arduino LilyPad USB se fait de la même manière que pour les autres cartes Arduino que tu as déjà utilisées : tu connectes la carte au câble USB et le câble USB à l'ordinateur.

Copyright © 2015 Eyrolles.

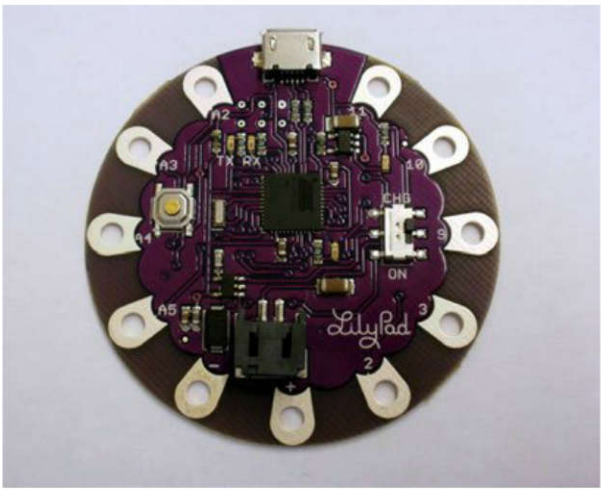


FIGURE 8-2 La carte Arduino LilyPad USB.

Les initiales FTDI signifient *Future Technology Devices International*, une société qui produit le circuit intégré qui est utilisé dans la plupart des cartes Arduino. Un circuit intégré FTDI traduit les signaux électriques transmis par l'ordinateur au travers du câble USB en signaux compréhensibles par le microcontrôleur Arduino. La carte Arduino Uno utilise un circuit intégré similaire au circuit FTDI qui permet au microcontrôleur de faire cette conversion. Il est déjà intégré à la carte Arduino Uno.

La carte Arduino LilyPad ne possède pas ce composant additionnel, il est donc nécessaire de la relier par l'intermédiaire d'une carte FTDI à chaque fois que l'on doit télécharger un sketch. L'Arduino LilyPad USB utilise un microcontrôleur différent de celui de la Uno ou de la LilyPad. Son circuit intégré assure la conversion des signaux USB sans l'aide d'un second circuit, tu n'auras donc pas à utiliser de carte FTDI.

Si tu es intéressé par les détails différenciant les cartes Arduino les unes des autres, regarde le guide comparatif (en anglais) à l'adresse `https://learn.sparkfun.com/tutorials/arduino-comparison-guide/introduction`.

Les autres types d'Arduino LilyPad ne possèdent pas de connecteur micro-USB. En revanche, ils contiennent six broches sur le haut de leur circuit imprimé. Ces broches permettent de les relier à une carte FTDI (voir la figure 8-3) qui possède un connecteur USB. Pour programmer ces cartes, tu dois d'abord connecter la carte FTDI à l'Arduino LilyPad, puis relier un câble USB à la carte FTDI, et enfin relier ce câble à ton ordinateur. Donc, si tu possèdes déjà une carte Arduino LilyPad et que ne trouves pas de carte Arduino LilyPad USB, ne t'inquiète pas ! Tu peux toujours utiliser une carte Arduino LilyPad différente, à la condition que tu possèdes également une carte FTDI.

Copyright © 2015 Eyrolles.

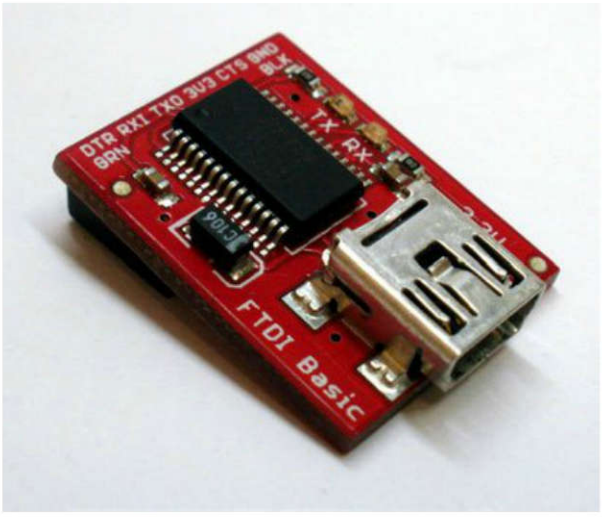


FIGURE 8-3 La carte de programmation FTDI dont tu as besoin si tu utilises une carte Arduino LilyPad différente de l'Arduino LilyPad USB.

Un clignotant utilisant une carte Arduino LilyPad

La première différence importante entre une carte Arduino Uno ou Leonardo et une Arduino LilyPad USB est la présence d'un interrupteur de mise en marche ! En effet, la carte ne s'allume pas automatiquement quand tu l'alimentes, ce qui peut être gênant et frustrant si tu n'es pas au courant de la présence de cet interrupteur. Tu pourrais même penser que ta carte est défectueuse ! Mais ne t'inquiète pas ! L'interrupteur est situé en haut de la carte, à l'opposé du bouton Reset par rapport au circuit intégré microcontrôleur. La figure 8-4 te montre sa position.

Ta carte Arduino LilyPad USB possède une fonction de recharge d'une batterie. C'est très utile pour les projets que l'on peut porter sur soi car ils nécessitent en général d'être alimentés par une batterie. L'interrupteur a deux positions : Marche et Charge (ON et CHG, respectivement). La position Charge sert à la recharge d'une batterie à partir du câble USB. Tu ne peux pas simultanément recharger la batterie et mettre le microcontrôleur en fonctionnement. Tu trouveras d'autres informations sur la façon d'alimenter ta carte un peu plus loin lors de la réalisation de ton vêtement.

Chaque fois que tu utilises une nouvelle carte Arduino, il est recommandé de commencer par l'envoi du message « Hello World » (« Bonjour tout le monde ») qui montre que ta carte fonctionne et que tu sais comment la programmer. Tu l'as déjà fait avec ta carte Arduino Uno dans l'aventure 1. Pour une carte Arduino, le programme « Bonjour tout le monde » est le sketch Blink.

Copyright © 2015 Eyrolles.

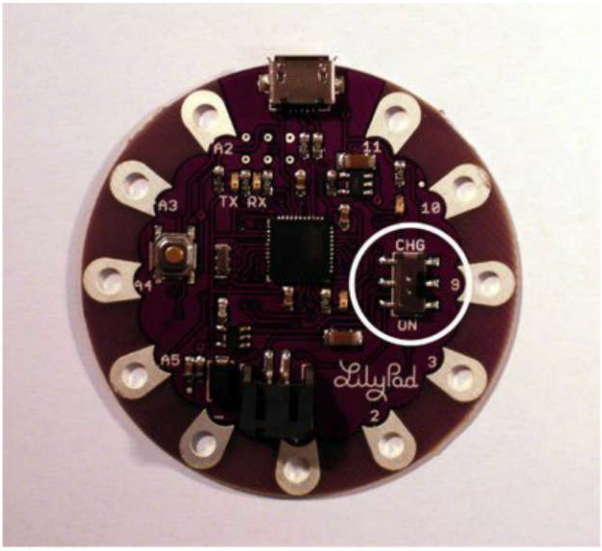


FIGURE 8-4 La position de l'interrupteur de mise en marche sur une Arduino LilyPad USB.

Ouvre le sketch Blink après avoir lancé l'IDE Arduino à partir du menu Fichier > Exemples > 01.Basics > Blink. Tu n'as rien à modifier dans ce sketch, tu peux immédiatement le télécharger dans ta carte Arduino LilyPad USB.

Pour le télécharger, tu dois relier la carte Arduino LilyPad USB à ton ordinateur et indiquer à l'IDE le type de carte utilisé dans le menu Outils > Type de carte > LilyPad Arduino USB, comme sur la figure 8-5. Clique ensuite sur le bouton Téléverser.

Une LED sur la carte LilyPad commence à clignoter : une seconde allumée, une seconde éteinte.

DÉFI

Les cartes Arduino nouvellement acquises sont souvent livrées avec un programme de clignotement déjà chargé, ta carte fait peut-être déjà clignoter sa LED reliée à la broche 13 pendant une seconde. Dans ce cas, modifie son comportement pour que la LED clignote pendant une demi-seconde au lieu d'une seconde. Téléverse le programme et vérifie que la LED clignote maintenant selon cette nouvelle séquence.

Copyright © 2015 Eyrolles.

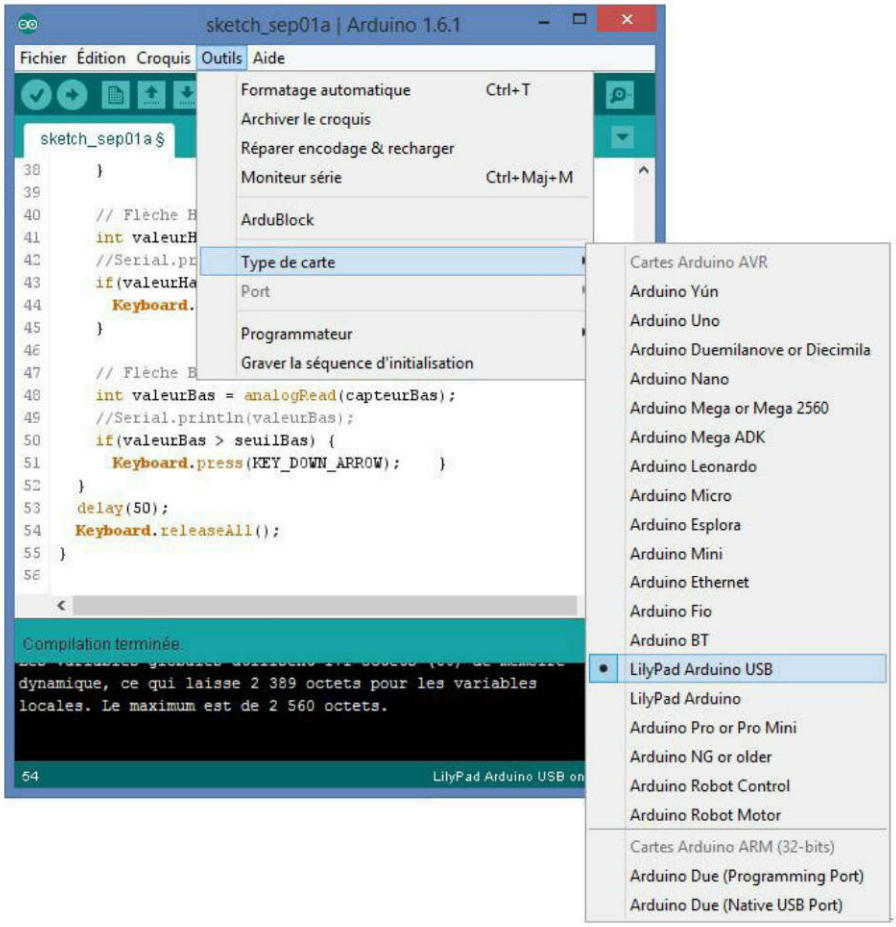


FIGURE 8-5 Choisis LilyPad Arduino USB dans la liste des cartes.

Prototypage des circuits souples

Maintenant que tu sais télécharger un sketch dans ta carte Arduino LilyPad USB, tu peux commencer à y relier des capteurs et des **actionneurs**. Ce qui est déroutant, c'est comment y parvenir, car cette carte LilyPad ne possède pas de broches dans lesquelles tu peux insérer des fils de pontage. Elle a de grandes pastilles trouées (plots) qui sont assez grandes pour y coudre facilement des cordelettes conductrices, réalisant ainsi un circuit souple. Mais tu ne veux peut-être pas coudre tous les circuits, car cela prend beaucoup de temps. Dès lors, comment réaliser le prototype d'un circuit souple ? La solution, c'est l'utilisation de câbles munis de pinces crocodiles.

Un **capteur** est un composant qui détecte quelque chose dans son environnement, comme de la lumière, un son ou un mouvement, et le traduit en un signal électrique. Les potentiomètres et les photorésistances sont ainsi des capteurs.

Copyright © 2015 Eyrolles.

Un **actionneur** est un élément qui transforme un signal électrique en une action dans son environnement, comme allumer une lampe ou produire un son ou un mouvement. Les moteurs, les lampes et les haut-parleurs en sont des exemples.

Les **circuits souples** sont des circuits réalisés avec des éléments souples tels que des fils ou des cordelettes conducteurs et du tissu. Ils sont souvent utilisés dans des projets de vêtements.

Des **câbles munis de pinces crocodiles** à leurs extrémités sont montrés sur la figure 8-6 : ce sont des fils munis d'attaches à ressorts qui ressemblent à la mâchoire d'un crocodile. Tu peux en relier une extrémité à ton Arduino LilyPad USB et l'autre à un élément de ton circuit, par exemple une LED.

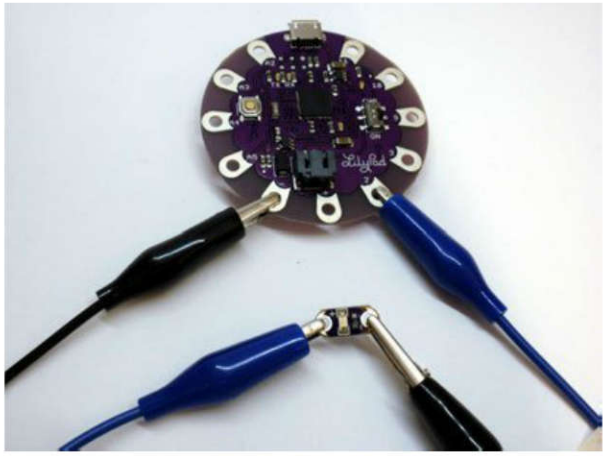


FIGURE 8-6 Utilise des câbles à pinces crocodiles au lieu de fils de pontage pour relier les composants utilisés dans un prototype de circuit souple.

Les **câbles à pinces crocodiles** sont des câbles équipés d'attaches à ressorts qui ressemblent à la mâchoire d'un crocodile. Ils sont utiles pour la réalisation de prototypes de circuits souples, pour relier des composants qui ne peuvent pas recevoir des fils de pontage.

Copyright © 2015 Eyrolles.

L'Arduino LilyPad USB fonctionne avec une tension inférieure à celle des cartes Arduino Uno et Leonardo. Quand une broche délivre un niveau `HIGH` (HAUT), la tension fournie est de 3,3 V au lieu de 5 V. Tu n'as pas à te soucier de la raison pour laquelle une tension de 3,3 V est utilisée au lieu de 5 V. En revanche, c'est important pour les LED, car elles ne nécessitent plus une résistance de limitation de courant de valeur aussi élevée. Tu peux utiliser une résistance comprise entre 68 et 100 Ω.

Pour tes projets de circuits souples, tu peux aussi acheter les LED LilyPad comme celles de la figure 8-7. Ce sont des LED montées sur une plaque de circuit imprimé mauve (de la même couleur que les cartes Arduino LilyPad) qui possèdent une résistance de limitation de courant intégrée à leur plaque. Leur avantage est de limiter le nombre de composants que tu dois coudre.

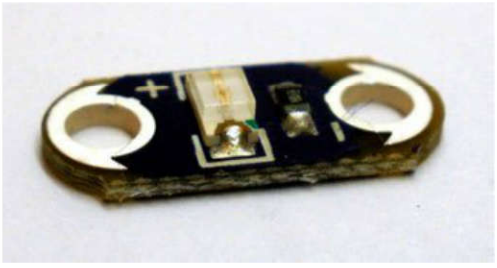


FIGURE 8-7 Les LED LilyPad possèdent une résistance de limitation de courant et peuvent être cousues.

DÉFI

Utilise des câbles à pinces crocodiles pour réaliser un circuit avec ton Arduino LilyPad USB et une LED. Si tu utilises une LED standard, tu dois utiliser une résistance de limitation de courant de 68 ou 100 Ω ; sinon, utilise une LED LilyPad.

Choisis un des plots pour relier ton circuit à LED. Tu peux remarquer que tu n'as pas autant de choix qu'avec les autres cartes Arduino. Tu peux choisir les plots 2, 3, 9, 10 ou 11, qui sont des connexions numériques. Choisis un plot et modifie le sketch Blink afin d'utiliser la sortie numérique choisie au lieu de la broche 13. Téléverse ton sketch et regarde ta LED clignoter !

Utiliser les tableaux astucieusement

Dans l'aventure 5, tu as appris à mémoriser une liste de valeurs dans un tableau sans avoir à créer une variable pour chaque valeur. Pour créer une liste d'entiers `int`, tu crées une variable de type de donnée `int[]`, puis tu fournis la liste des valeurs entre des accolades `{ }`, comme sur la figure 8-8.

Copyright © 2015 Eyrolles.

Les listes sont des tableaux à une dimension, mais les tableaux peuvent avoir plus d'une dimension. Ils peuvent en avoir le nombre que tu veux, même s'il est difficile de représenter quelque chose avec plus de trois dimensions. Tu as probablement déjà utilisé des tableaux à deux dimensions : ils sont organisés en lignes et colonnes comme dans un tableur. La figure 8-9 montre une façon de représenter un tableau à deux dimensions.

```
int listeEntiers [] = { 7, 9, 5, 1, 9, 2} ;
listeEntiers[1] est la valeur 9
listeEntiers[5] est la valeur 2
```

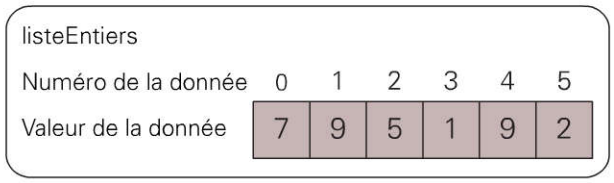


FIGURE 8-8 Une liste de nombres entiers également appelée « tableau à une dimension ».

Un **tableau à deux dimensions** est un ensemble de données organisé en lignes et en colonnes comme dans un tableur.

```
Int tableauEntiers[] = {{7, 9, 5, 1, 9, 2},
                        {9, 0, 3, 3, 2, 2},
                        {5, 4, 2, 7, 8, 3}} ;

tableauEntiers[1] [2] est la valeur 4
tableauEntiers[0] [5] est la valeur 2
```

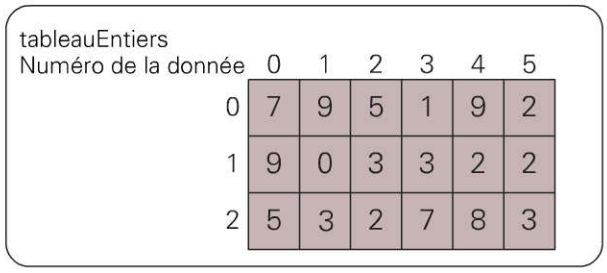


FIGURE 8-9 Un tableau à deux dimensions mémorise des données arrangées en lignes et en colonnes.

Tu vas utiliser des tableaux à deux dimensions pour mémoriser les messages que ton vêtement va afficher. Chaque lettre de ton message sera un tableau. Regarde un panneau d'affichage à LED d'un arrêt de bus ou d'une gare, qui affiche des lettres et des nombres. Tu peux représenter chaque caractère dans un rectangle, chaque lettre ou chiffre ayant une hauteur et une largeur. Dans ce rectangle, les points lumineux sont allumés selon un schéma représentant la lettre ou le chiffre. Ces

Copyright © 2015 Eyrolles.

rectangles de LED pour chaque caractère sont des tableaux à deux dimensions. Créer un tableau à deux dimensions demande un peu plus de travail que pour un tableau à une dimension, car tu n'as pas à indiquer le nombre d'éléments composant une liste. Tu dois simplement les lister entre les accolades { } et l'IDE Arduino les comptera pour toi. Mais pour créer un tableau à deux dimensions, tu dois compter toi-même les valeurs qu'il contiendra.

Le code suivant crée une variable appelée `Tableau2D` qui a trois colonnes et deux lignes :

```
int tableau2D [2] [3] = {{1,2,3},
                         {4,5,6}} ;
```

Tu peux mémoriser le nombre de lignes et de colonnes dans des variables, car ce sont des informations utiles auxquelles tu pourras faire référence par la suite. Si tu le fais, tu dois ajouter `const` en tête des variables. Cela signifie que leurs valeurs ne pourront pas être modifiées.

```
const int nombreRangees = 2 ;
const int nombreColonnes = 3 ;
int tableau2D [nombreRangees] [nombreColonnes] = {{1,2,3},
                                                  {4,5,6}} ;
```

Tu n'as peut-être pas encore bien réalisé pourquoi tu pourrais avoir besoin de tableaux à deux dimensions, alors construisons un circuit qui les utilise. Réalise le circuit de la figure 8-10 avec des LED reliées aux plots 2, 3, 9, 10 et 11. Commence par relier chaque plot de ton Arduino LilyPad USB au côté positif de chaque LED. Puis relie tous les plots négatifs des LED ensemble. Connecte ensuite le plot négatif de la LED reliée à la sortie 2 au plot de masse GND de ta carte Arduino LilyPad USB.

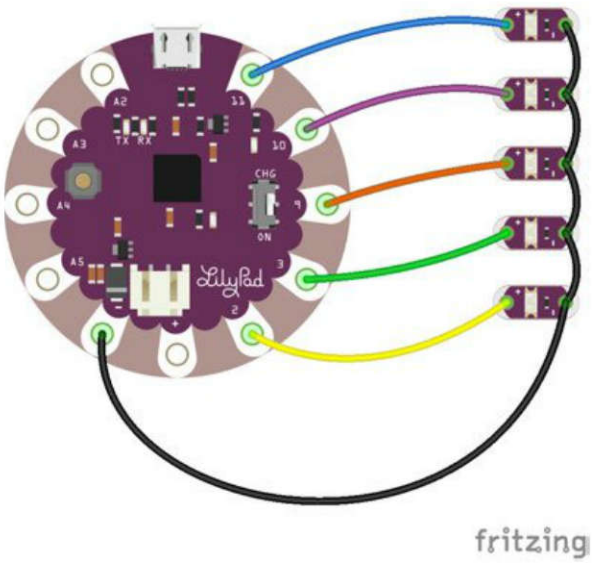


FIGURE 8-10 Le circuit pour l'utilisation d'un tableau d'état des LED.

Copyright © 2015 Eyrolles.

Si tu ne possèdes pas de LED LilyPad, remplace-les par des LED normales et des résistances de limitation de courant.

Tu vas maintenant animer les LED et mémoriser chaque figure de l'animation dans un tableau à deux dimensions. Commence par ouvrir l'IDE Arduino et crée un nouveau sketch. Entre ensuite les fonctions `setup()` et `loop()` en les laissant vides :

```
void setup() {

}

void loop() {

}
```

Crée deux variables qui mémoriseront le nombre de lignes et de colonnes du tableau au début du sketch. Le tableau contient une colonne pour chaque LED (soit cinq colonnes) et une ligne pour chaque trame de l'animation. Celle de cet exemple a neuf trames (donc neuf lignes) :

```
const int nombreLED = 5 ;
const int nombreTrames = 9 ;
```

Crée ensuite le tableau qui va mémoriser chacun des numéros des plots reliés aux LED :

```
int plotsLED [] = {2, 3, 9, 10, 11} ;
```

Puis remplis le tableau à deux dimensions de l'animation :

```
int trames [nombretrames] [nombreLED] =
{
{1, 0, 0, 0, 0},
{0, 1, 0, 0, 0},
{0, 0, 1, 0, 0},
{0, 0, 0, 1, 0},
{0, 0, 0, 0, 1},
{0, 0, 0, 1, 0},
{0, 0, 1, 0, 0},
{0, 1, 0, 0, 0},
{1, 0, 0, 0, 0}
};
```

Copyright © 2015 Eyrolles.

Ajoute les lignes de code suivantes dans `setup()` afin de préciser le mode de chaque plot des LED :

```
int i;
for(int i=0; i<nombreLED; i++) {
  pinMode(plotsLED[i], OUTPUT);
}
```

Dans `loop()`, commence par le code avec lequel tu n'es pas familiarisé. Pour l'instant, ajoute le code à ton sketch, tu l'étudieras en détail plus loin :

```
// les variables pour mémoriser la trame et la LED en cours
int trame;
int led;

for( trame=0; trame<nombreTrames; trame++ ) {
  // itération pour chaque trame mémorisée dans une colonne
  for( led=0; led<nombreLED; led++ ) {
    // allumer ou éteindre chaque LED de la trame
    digitalWrite(plotsLED[led], trames[trame][led] );
  }
  delay(300); // pause entre chaque trame
}
```

Tu as déjà utilisé des boucles `for` précédemment pour accéder à chaque élément d'une liste un par un. Puisque le tableau `trames` possède deux dimensions (imagine cela comme étant une liste de listes), deux boucles `for` sont nécessaires.

La première parcourt les colonnes qui contiennent une trame de données décrivant ce que doit être l'état de chaque LED. La seconde parcourt la trame et allume ou éteint chaque LED. Une fois que la seconde boucle `for` a allumé ou éteint chaque LED, le sketch fait une pause avant de continuer avec la trame suivante. Sans cette pause, l'animation serait trop rapide pour être observée ! Tu peux voir, sur la figure 8-11, comment le tableau est parcouru ligne par ligne.

Le code suivant est le sketch complet. Construis le circuit et télécharge le sketch pour voir l'animation en mouvement :

```
const int nombreLED = 5 ;
const int nombreTrames = 9 ;

// Les numéros des plots reliés aux LED
int plotsLED [] = {2, 3, 9, 10, 11} ;
```

Copyright © 2015 Eyrolles.

```
// Les trames de l'animation
int trames [nombreTrames] [nombreLED] =
{
{1, 0, 0, 0, 0},
{0, 1, 0, 0, 0},
{0, 0, 1, 0, 0},
{0, 0, 0, 1, 0},
{0, 0, 0, 0, 1},
{0, 0, 0, 1, 0},
{0, 0, 1, 0, 0},
{0, 1, 0, 0, 0},
{1, 0, 0, 0, 0}
};

void setup() {
  // Indiquer le mode de fonctionnement des plots
  int i;
  for(int i=0; i<nombreLED; i++) {
    pinMode( plotsLED[i], OUTPUT);
  }
}

void loop() {
  // Les variables pour mémoriser la trame et la LED en cours
  int trame;
  int led;

  for( trame=0; trame<nombreTrames; trame++ ) {
    // itération pour chaque trame mémorisée dans une colonne
    for( led=0; led<nombreLED; led++ ) {
      // allumer ou éteindre chaque LED de la trame
      digitalWrite( plotsLED[led], trames[trame][led] );
    }
    delay(300); // pause entre chaque trame
  }
}
```

Copyright © 2015 Eyrolles.

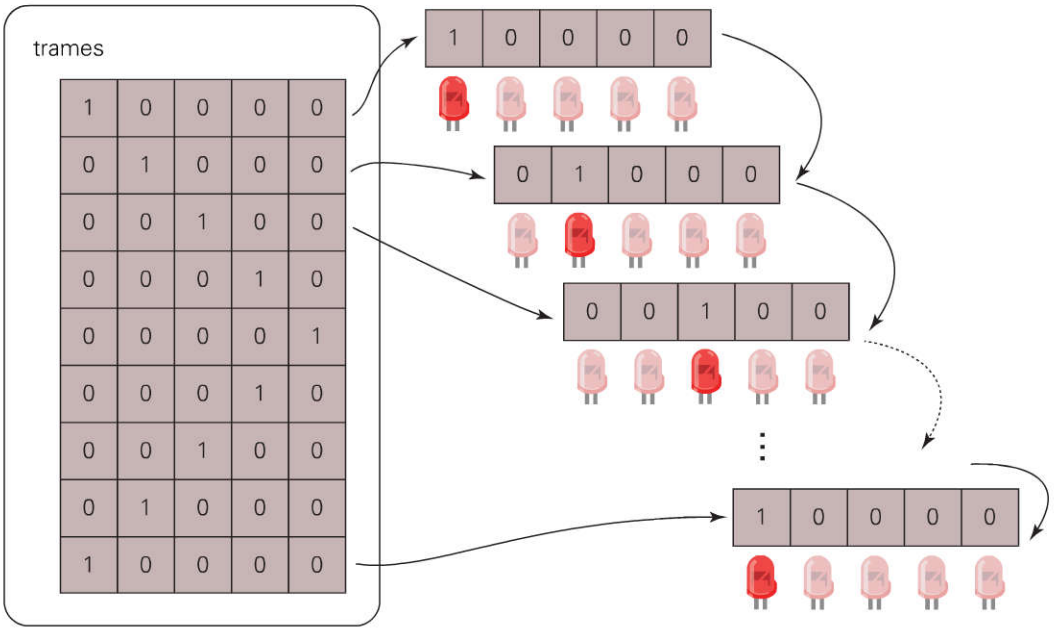


Tableau complet à deux dimensions nommé trames.

Chaque ligne précise comment les cinq LED doivent être allumées ou éteintes.

FIGURE 8-11 Parcours des trames d'une animation mémorisée dans un tableau à deux dimensions.

Passage de données entre fonctions

Si tu souhaites faire autre chose en plus de cette animation dans la fonction `loop()`, il vaudra mieux placer le code de l'animation dans sa propre fonction. L'exemple suivant te montre comment.

Crée une fonction appelée `affichageAnimation()`, puis coupe et colle le code de l'animation depuis la fonction `loop()` :

```
void affichageAnimation() {
  // les variables pour mémoriser la trame et la LED en cours
  int trame;
  int led;
  for( trame=0; trame<nombreTrames; trame++ ) {
    // itération pour chaque trame mémorisée dans une colonne
    for( led=0; led<nombreLED; led++ ) {
      // allumer ou éteindre chaque LED de la trame
      digitalWrite( plotsLED[led], trames[trame][led] );
    }
    delay(300); // pause entre chaque trame
  }
}
```

Copyright © 2015 Eyrolles.

Rappelle-toi que tu dois maintenant appeler la nouvelle fonction depuis la fonction `loop()`, sinon l'animation ne sera jamais exécutée. La nouvelle fonction `affichageAnimation()` doit se situer après la fonction `loop()`, qui doit maintenant comporter le code suivant :

```
void loop() {
  affichageAnimation() ;
}
```

Maintenant que le code contrôlant l'animation est contenu dans une fonction, tu peux en profiter pour étudier son fonctionnement. Tu peux transmettre des données à une fonction afin qu'elle change son comportement selon les données qu'elle reçoit.

Pour transmettre des données, tu dois créer un argument. Cela se fait dans la ligne de code où tu indiques le nom de la fonction. Tu dois également donner un nom à cet argument et indiquer son type de donnée.

Change ta fonction de la façon suivante :

```
void affichageAnimation(int vitesseAnimation) {}
```

La fonction `affichageAnimation()` possède maintenant un argument qui est un entier `int`, qui se nomme `vitesseAnimation` et qui va contrôler le temps de pause de l'animation entre chaque trame. Utilise cette variable pour préciser le temps de la fonction `delay()`.

Change la ligne de code qui définit la durée de la pause entre chaque trame de la façon suivante :

```
delay(vitesseAnimation) ; // Pause entre chaque trame
```

Tu peux maintenant appeler la fonction `affichageAnimation()` avec des valeurs différentes de l'argument `vitesseAnimation`, pour exécuter l'animation à des vitesses différentes ! Le programme suivant représente le sketch complet. Télécharge-le dans ta carte Arduino LilyPad USB avec les cinq LED reliées et observe l'animation décrite par le tableau à deux dimensions :

```
// const indique que les valeurs ne changeront pas !
const int nombreLED = 5 ;
const int nombreTrames = 9 ;

// Les numéros des plots reliés aux LED
int plotsLED [] = {2, 3, 9, 10, 11} ;

// Les trames de l'animation
int trames [nombreTrames] [nombreLED] =
{
```

Copyright © 2015 Eyrolles.

```
{1, 0, 0, 0, 0},
{0, 1, 0, 0, 0},
{0, 0, 1, 0, 0},
{0, 0, 0, 1, 0},
{0, 0, 0, 0, 1},
{0, 0, 0, 1, 0},
{0, 0, 1, 0, 0},
{0, 1, 0, 0, 0},
{1, 0, 0, 0, 0}
};

void setup() {
  // Indiquer le mode de fonctionnement des plots
  int i;
  for(int i=0; i<nombreLED; i++) {
    pinMode( plotsLED[i], OUTPUT);
  }
}

void loop() {
  affichageAnimation( 100 );
  affichageAnimation( 500 );
  affichageAnimation( 1000 );
}

void affichageAnimation( int vitesseAnimation) {
  // variables pour mémoriser la trame et la LED en cours
  int trame;
  int led;

  for( trame=0; trame<nombreTrames; trame++ ) {
    // itération pour chaque trame mémorisée dans une colonne
    for( led=0; led<nombreLED; led++ ) {
      // allumer ou éteindre chaque LED de la trame
      digitalWrite( plotsLED[led], trames[trame][led] );
    }
    delay(vitesseAnimation); // pause entre chaque trame
  }
}
```

Copyright © 2015 Eyrolles.

DÉFI

Réalise le circuit pour l'animation des LED présenté dans la section « Passage de données entre fonctions » page 270. Modifie le code du sketch en utilisant une boucle `for` pour répéter l'exécution des animations à des vitesses allant de 50 à 500.

Rappelle-toi que tu peux incrémenter la vitesse avec des valeurs supérieures à un, par exemple en créant la boucle `for` suivante :

```
int i;
for( i=0; i<100; i+=5) {

}
```

Réalisation d'un vêtement à persistance de vision

As-tu déjà agité un cierge magique et remarqué que les étincelles semblaient former un trait lumineux ? Ou alors, peut-être as-tu déjà vu un jouet possédant plusieurs LED qui affichaient un mot ou une image quand on l'agitait rapidement ? Ce sont deux exemples qui utilisent la persistance rétinienne de la vision (POV ou *Persistence Of Vision*) dans lesquels tes yeux et ton cerveau continuent de voir une trace lumineuse après que la source lumineuse a bougé ou est éteinte. Dans le cas du cierge magique, les étincelles rougeoyantes se déplacent, mais ton cerveau continue de les voir à l'endroit où elles étaient, comme une traînée. En ce qui concerne le jouet, les LED clignotent rapidement selon un schéma précis. Ton cerveau assemble les différents schémas quand le jouet est déplacé et les interprète comme un mot ou une image.

Ce qui est excitant, c'est que tu vas pouvoir créer ton propre affichage à persistance rétinienne avec ton Arduino LilyPad USB. Et puisque tu utilises une LilyPad, tu vas pouvoir coudre ton circuit sur un vêtement et le porter. Tu pourras alors programmer ta LilyPad USB afin d'afficher un message secret qui pourra être visualisé en prenant une photo en pause prolongée (voir la figure 8-12). Tu peux coudre ton circuit sur la manche d'un sweat-shirt, mais aussi sur n'importe quel autre vêtement !

FIGURE 8-12 Un vêtement à persistance de vision rétinienne.

Copyright © 2015 Eyrolles.

Ce dont tu as besoin

- Un ordinateur ;
- une carte Arduino LilyPad USB ;
- un câble micro-USB ;
- 14 câbles à pinces crocodiles ;
- 7 LED LilyPad (ou 7 LED standards et 7 résistances de 68 Ω ou 100 Ω si tu n'utilises pas de LED LilyPad) ;
- une batterie Lithium Ion Polymère (LiPo) ;
- du fil à coudre standard ;
- du fil à coudre (cordelette) conducteur d'électricité ;
- un vêtement ;
- une aiguille à coudre ;
- des ciseaux ;
- de la colle blanche ;
- une paire de pinces plates (si tu n'utilises pas de LED LilyPad).

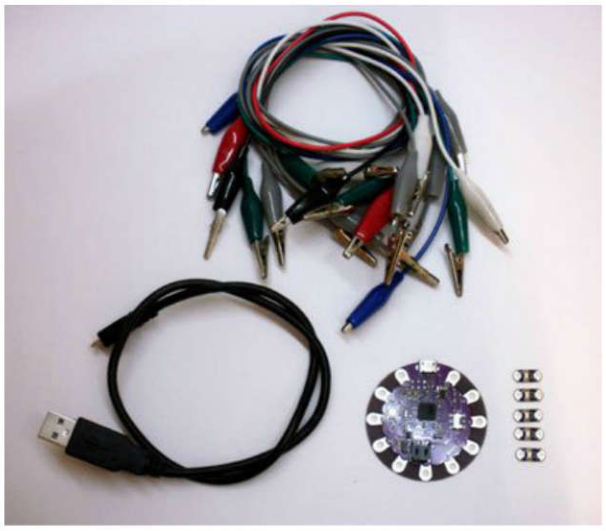


FIGURE 8-13 Les composants électroniques dont tu as besoin pour le vêtement à persistance de vision.

Comprendre le circuit

Le circuit pour le vêtement à vision persistante est similaire à celui de l'animation des LED que tu as réalisé auparavant dans cette aventure. La seule différence est l'utilisation de 7 LED au lieu de 5. Le schéma de ce circuit se trouve sur la figure 8-14.

Copyright © 2015 Eyrolles.

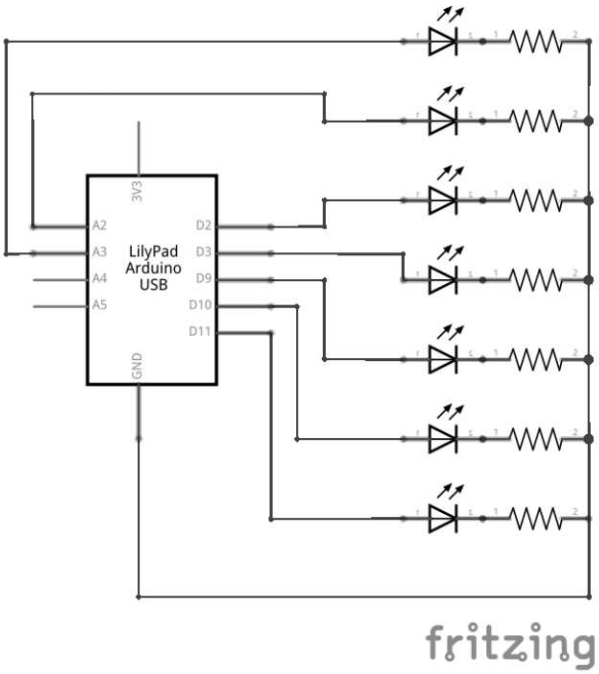


FIGURE 8-14 Le schéma du circuit du vêtement à persistance de vision rétinienne.

Prototypage avec des câbles à pinces crocodiles

Le circuit du vêtement étant un circuit souple, tu dois réaliser son prototype à l'aide de câbles à pinces crocodiles, car la carte Arduino LilyPad USB et les LED LilyPad ne peuvent pas s'insérer dans une plaque de montage. Réalise le circuit de la figure 8-15.

Commence par relier chaque plot de la LilyPad USB au plot positif de chaque LED LilyPad. Relie ensuite les plots négatifs des LED LilyPad ensemble. Connecte le plot négatif de la LED reliée au plot 2 de la LilyPad USB au plot de masse GND.

Si tu ne disposes pas de LED LilyPad, remplace-les par des LED standards et des résistances de limitation de courant.

Copyright © 2015 Eyrolles.

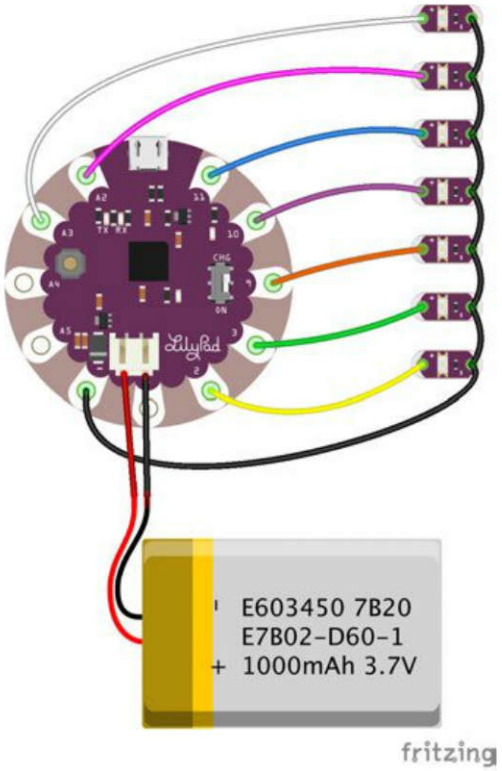


FIGURE 8-15 Le prototype du circuit réalisé avec des câbles à pinces crocodiles.

Charger la batterie

La carte Arduino LilyPad USB possède un connecteur prévu pour la recharge d'une batterie lithium ion polymère (LiPo) ainsi qu'un circuit chargeur incorporé. Les batteries LiPo sont plates et de couleur argentée. Tu n'as pas besoin d'une batterie de grande capacité pour ce projet. La capacité des batteries se mesure en milliampères-heures (mAh), qui indiquent la quantité de courant pouvant être fourni en un temps donné. Une batterie de 500 mAh est un bon compromis – elle n'est pas trop grosse et ne doit pas être rechargée trop fréquemment –, mais tu peux en utiliser une de capacité différente que tu pourras acheter facilement.

Les batteries LiPo peuvent être dangereuses si tu n'en prends pas soin correctement. Ne les laisse pas sans surveillance pendant leur recharge ! Je te conseille de les acheter chez un fournisseur de confiance comme Adafruit ou SparkFun ou l'un de ses distributeurs. N'achète que des batteries LiPo qui possèdent un circuit de protection et n'en utilise jamais une qui te semble gonflée ou endommagée. Adafruit propose un guide d'utilisation des batteries LiPo (en anglais) à l'adresse `https://learn.adafruit.com/li-ion-and-lipoly-batteries`.

Copyright © 2015 Eyrolles.

Pour recharger la batterie, relie-la à ta carte Arduino LilyPad USB, que tu connecteras elle-même à une source d'alimentation ou à ton ordinateur par l'intermédiaire de son câble micro-USB. Assure-toi que l'interrupteur de la LilyPad USB soit bien dans la position charge : la LED au-dessus de l'inscription CHG de la carte doit être allumée. Le rechargement de la batterie s'arrête automatiquement lorsqu'elle est totalement chargée.

Écriture du programme

Le code pour afficher le message en vision persistante est similaire à celui de l'animation avec des LED étudiée au début de cette aventure. Les trames pour les LED sont mémorisées dans des tableaux bidimensionnels : un pour chaque caractère et un pour l'espace. Étant donné le nombre important de tableaux nécessaires, et afin de mieux organiser le programme, ils sont tous situés dans un fichier en-tête (.h). Le code est disponible à l'adresse `www.editions-eyrolles.com/go/arduino`.

Télécharge le fichier `vetement_pov` de l'aventure 8 et ouvre-le dans l'IDE Arduino. Le sketch doit posséder deux onglets : l'un appelé `vetement_pov` et l'autre `alphabet.h`.

Clique sur l'onglet `vetement_pov`. Change le message en gras dans la fonction `loop()` par le message que tu veux afficher avec ton vêtement, puis télécharge le sketch dans ta carte Arduino LilyPad USB. Ton message ne peut comporter que des lettres majuscules et des espaces.

```
void loop()
{
      String message = "HELLO    "; /* indiquer ici un
  message en majuscules */
  afficheTexte(message);
}
```

Déplace ton circuit en un mouvement de vagues (ce sera plus facile si un ami le déplace dans une pièce obscure) ou prends une photographie en pause prolongée pour voir le message apparaître, comme sur la figure 8-16.

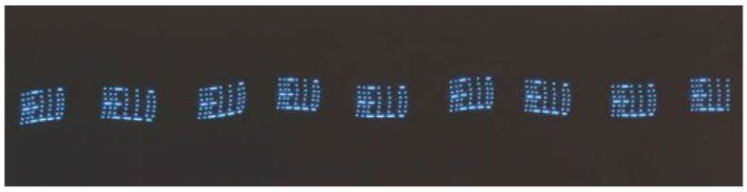


FIGURE 8-16 Le message en vision persistante capturé par un appareil photographique en pause prolongée.

Copyright © 2015 Eyrolles.

EXPLORATION DU CODE

Ouvre le sketch que tu viens de télécharger dans l'IDE Arduino et clique sur l'onglet `alphabet.h` pour voir le code du fichier en-tête.

Le code suivant est celui de la lettre A. Si tu tournes ta tête de côté, tu pourrais voir la forme de la lettre A représentée par les 1 :

```
int A[nombreTrames][nombreLED] = {
  {1,1,1,1,1,1,0},
  {0,0,1,0,0,0,1},
  {0,0,1,0,0,0,1},
  {0,1,1,0,0,1,1},
  {1,1,1,1,1,1,0}
};
```

Dans le sketch principal, le fichier en-tête qui décrit toutes les lettres est importé au tout début du fichier :

```
#include "alphabet.h"
```

Il y a ensuite trois variables qui décrivent le temps pendant lequel les LED sont éteintes entre deux lettres, le temps où elles affichent la trame d'une lettre et les numéros des plots des LED. La carte Arduino LilyPad USB n'ayant que cinq plots numériques, deux sorties analogiques sont également utilisées.

```
int LEDeteintes = 9; /* temps d'extinction des LED entre
  deux lettres */
int LEDallumee = 3; // temps d'allumage des LED
int brochesLED[] = {2, 3, 9, 10, 11, A2, A3};
```

Dans la fonction `setup()`, chaque appel à `pinMode()` prépare les plots en mode sortie (`OUTPUT`) et la communication série est initialisée. La communication série est utilisée pour la mise au point afin de voir ce qui est généré en sortie sur le moniteur série :

```
// définition du mode des broches à OUTPUT
int i;
for(int i=0; i<nombreLED; i++) {
  pinMode( brochesLED[i], OUTPUT);
}

Serial.begin(9600);
```

La fonction `loop()` est assez simple, car la majeure partie des autres fonctions réalisent les actions. Le message à afficher est mémorisé dans la variable

Copyright © 2015 Eyrolles.

`message` puis il est transmis comme argument à la fonction `afficheTexte()`. Le fichier en-tête ne décrivant que des lettres majuscules, le message doit être écrit en lettres majuscules et il ne doit comporter ni signe de ponctuation, ni lettres minuscules.

```
String message = "HELLO    "; /* indiquer ici un message
  en majuscules */
afficheTexte(message);
```

La fonction `afficheTexte()` est assez longue, mais elle ne réalise que des actions simples plusieurs fois. Elle lit le message lettre par lettre, et indique ensuite à `afficheLettre()` d'afficher cette lettre du message. Le code suivant n'est que pour les lettres A, B et C, cela te donne une idée de la façon dont le reste de l'alphabet fonctionne :

```
for (int i=0; i<texte.length(); i++)
{
  switch(texte[i])
  {
    case 'A' :
      afficheLettre(A);
      break;
    case 'B' :
      afficheLettre(B);
      break;
    case 'C' :
      afficheLettre(C);
      break;
  }
}
```

La fonction `afficheLettre()` est identique à la fonction `afficheAnimation()` que tu as écrite précédemment. Elle parcourt chaque trame de la lettre qui est affichée. Elle affiche également ce qui est envoyé aux LED sur le moniteur série pour t'aider à voir ce qui se passe. À la fin de la fonction se trouve une boucle `for` supplémentaire qui permet de faire une pause entre chaque lettre. Sinon, il serait difficile de lire les lettres une par une, car elles seraient floues :

```
int trame;
int led;

// affichage de la lettre
for( trame=0; trame<nombreTrames; trame++ ) {
  for( led=0; led<nombreLED; led++ ) {
    digitalWrite( brochesLED[led], lettre[trame][led] );
    Serial.print(lettre[trame][led]);
```

Copyright © 2015 Eyrolles.

```
    }
    Serial.println();
    // délai entre l'affichage de chaque colonne
    delay(LEDallumee);
  }
  Serial.println("-------");
  // afficher un espace après chaque lettre
  for( led=0; led<nombreLED; led++ ) {
    digitalWrite( brochesLED[led], 0 );
  }
  // délai pour l'espace entre lettres
  delay(LEDeteintes);
```

Fabrication de ton vêtement à vision persistante

Après avoir réalisé le prototype de ton circuit à l'aide de câbles à pinces crocodiles et téléchargé le sketch afin de vérifier que tout fonctionnait, te voilà prêt à faire de la couture ! En utilisant du fil à coudre conducteur, tu vas coudre les plots de ton Arduino LilyPad USB aux plots de chacune des LED. Rappelle-toi que le fil à coudre conducteur est identique à du fil électrique, mais qu'il ne possède pas de gaine isolante pour le protéger. Cela signifie que si deux morceaux de fil conducteur se touchent, puisqu'ils conduisent l'électricité, ils vont provoquer un court-circuit. Utilise un morceau de fil différent pour chaque connexion et fais bien attention à ce qu'ils ne touchent pas accidentellement une autre partie du circuit.

Coudre des LED

Si tu n'as pas de LED LilyPad, tu peux toujours utiliser des LED prévues pour les plaques de montage. En utilisant tes pinces plates, plie les fils de tes LED et de tes résistances afin de former des boucles au travers desquelles tu peux coudre un fil. Tu peux plier différemment le fil long et le fil court de tes LED pour pouvoir repérer la boucle du fil positif et celle du fil négatif. J'aime plier le fil positif en cercle et le fil négatif en carré (parce qu'il me rappelle le signe moins), comme sur la figure 8-17, mais tu peux les plier selon les formes que tu préfères pour les repérer.

FIGURE 8-17 Le pliage des fils des composants pour leur permettre d'être cousus.

Copyright © 2015 Eyrolles.

Coudre ton électronique

Couds le circuit en procédant selon les étapes suivantes.

1. Commence par choisir l'endroit où tu vas placer tes LED sur la manche droite de ton vêtement. Tu peux marquer leur emplacement avec des épingles ou de la craie.
2. Commence par la LED du bas. Utilise du fil conducteur, couds le plot 9 au plot positif de la LED. Maintiens le fil fermement sur le tissu du vêtement et couds le plot de ton Arduino LilyPad USB avec le plot de la LED LilyPad en prévoyant environ cinq points pour les maintenir sur le tissu. Fais un nœud et coupe ton fil. Regarde la figure 8-18 pour te guider.

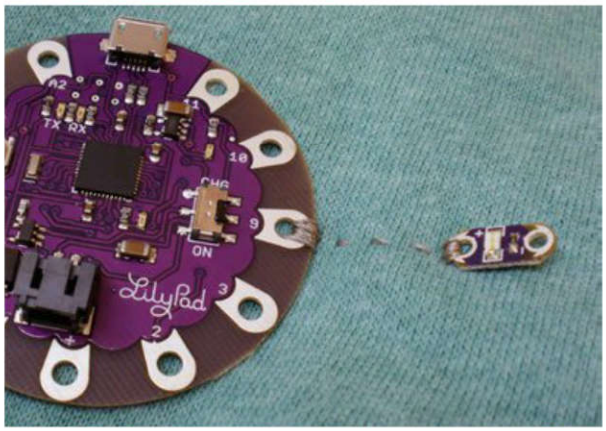


FIGURE 8-18 La première connexion de couture du circuit LilyPad.

3. De façon identique, si tu utilises des LED standards avec des résistances, couds le plot d'Arduino à la LED. Puis, en utilisant un morceau de fil conducteur différent, couds la LED à la résistance.
4. Avec une longueur de fil conducteur différent, couds le plot 10 au côté positif de la LED suivante juste au-dessus de celle que tu viens de coudre. À nouveau, couds fermement les plots au tissu puis fais un nœud et coupe le fil.
5. Continue ainsi avec chaque LED restante. Fais bien attention que les fils des coutures ne se touchent pas (voir la figure 8-19).
6. Les plots négatifs des LED (ou des résistances, si tu n'utilises pas de LED LilyPad) peuvent tous se toucher. Tu peux donc les coudre avec une seule longueur de fil conducteur. Commence par le plot négatif de ta carte Arduino LilyPad USB que tu vas coudre au plot négatif de la LED du bas. Après avoir bien cousu le plot négatif de cette LED, continue ta couture jusqu'à la LED suivante.
7. Répète l'étape 6 pour coudre le plot négatif de la LED suivante jusqu'à ce que tu aies cousu toutes les LED comme sur la figure 8-20.
8. Fixe tous les nœuds avec un peu de colle blanche. Comme ils peuvent parfois se défaire avec le temps, la colle permet d'éviter cet inconvénient.

Copyright © 2015 Eyrolles.

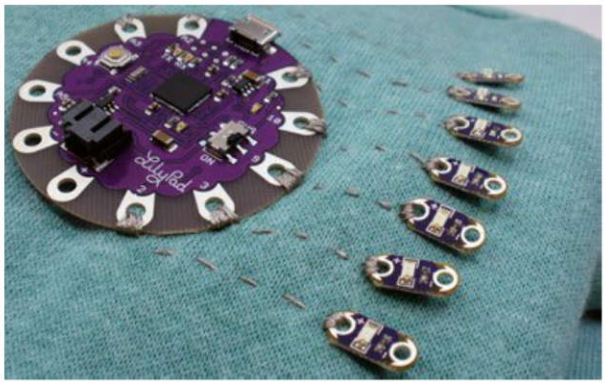


FIGURE 8-19 Couture des LED suivantes au circuit.

9. En utilisant du fil à coudre standard, ajoute une couture autour de la batterie pour la maintenir attachée à ton vêtement.

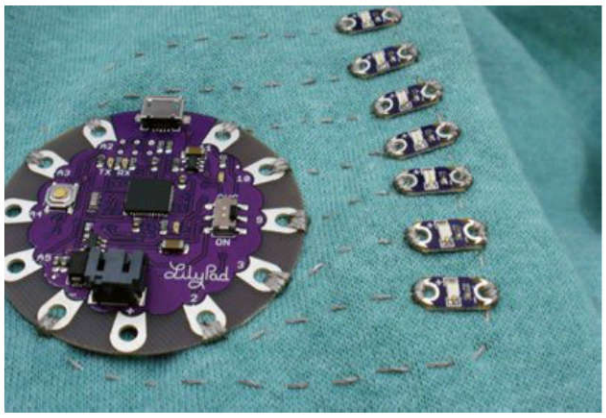


FIGURE 8-20 Le circuit du vêtement à vision persistante entièrement cousu.

Tu n'as peut-être pas envie de coudre ta carte Arduino LilyPad USB directement sur ton vêtement, surtout si tu veux utiliser la même carte pour différents projets. La carte Arduino LilyPad SimpleSnap (`http://arduino.cc/en/Main/ArduinoLilyPadSimpleSnap`) comporte des boutons pression femelles soudés à chaque plot. Tu peux alors coudre des boutons pression mâles leur correspondant sur ton vêtement, en utilisant du fil à coudre conducteur. Les boutons pression étant en métal, ils conduisent l'électricité tout en te permettant de retirer ta carte de ton vêtement sans l'endommager.

Tu peux modifier ta carte Arduino LilyPad USB de la même façon en y soudant des boutons pression toi-même. Tu peux trouver ce type de boutons dans n'importe quelle mercerie.

Copyright © 2015 Eyrolles.

Plus d'aventures avec l'Arduino LilyPad

Si tu veux en savoir plus sur l'Arduino LilyPad USB, rends-toi sur le site d'Arduino à l'adresse `http://arduino.cc/en/Guide/ArduinoLilyPadUSB`.

Tu peux également en apprendre davantage sur la façon d'utiliser les cartes LilyPad et trouver d'autres projets sur le site `http://lilypadarduino.org`.

Il existe de nombreux livres en anglais traitant des circuits souples que tu pourrais lire. En voici quelques-uns :

- *Fashioning Technology* par Syuzi Pakhchyan (Maker Media, 2008)
- *Make: Wearable Electronics* par Kate Hartman (Maker Media, 2014)
- *Switch Craft* par Alison Lewis et Fang-Yu Lin (Potter Craft, 2008)
- *Sew Electric* par Leah Buechley, Kanjun Qi et Sona de Boer (HLT Press, 2013)

Rappel des commandes Arduino de cette aventure

Commande	Description
`int [] []`	Indique que la variable est un tableau à deux dimensions contenant des valeurs stockées dans des lignes et des colonnes.
`const`	Indique que la variable est une constante dont la valeur ne peut pas être modifiée.

Copyright © 2015 Eyrolles.

Étape franchie : Tu es désormais un brillant spécialiste des vêtements Arduino et des variables à multiples dimensions !

Dans la prochaine aventure

Tu mettras à profit toutes tes connaissances pour créer un jeu inspiré du flipper !

Copyright © 2015 Eyrolles.

Aventure 9
La grande aventure : un labyrinthe à bille

TU AS PARCOURU UN LONG CHEMIN depuis le début de ce livre, et te voilà arrivé à l'aventure finale, la plus passionnante de toutes ! Dans cette aventure, tu vas combiner les entrées et sorties numériques et les entrées et sorties analogiques pour créer un labyrinthe à bille comme celui de la figure 9-1. Ce jeu s'inspire des anciens jeux de flipper, il saura quand tu marqueras des points et te dira si tu as atteint un nouveau score record. Il décomptera le temps de jeu depuis le début d'une nouvelle partie et suivra le temps qui reste avant la fin de la partie. Il possédera même ses propres effets sonores !

Le jeu combine beaucoup des connaissances que tu as apprises dans les aventures précédentes, mais il y a aussi quelques autres sujets que tu dois connaître avant de commencer à le construire. Le premier concerne l'utilisation d'un élément piézoélectrique comme capteur. Tu en as déjà utilisé un comme haut-parleur, mais il s'agit d'un composant aux multiples possibilités qui te permet aussi de détecter des vibrations, ce qui est essentiel pour ton labyrinthe à bille !

Copyright © 2015 Eyrolles.

FIGURE 9-1 Le jeu du labyrinthe à bille de la grande aventure, une fois terminé.

Ce dont tu as besoin

Tu as besoin des éléments suivants pour réaliser ton labyrinthe. Les composants électroniques nécessaires sont présentés sur la figure 9-2 :

- Un ordinateur ;
- une carte Arduino Uno ;
- un câble USB ;
- une plaque de montage ;
- des fils de pontage ;
- 6 éléments piézoélectriques ;
- 6 résistances d'1 MΩ ;
- une LED rouge ;
- une LED jaune ;
- une LED verte ;
- 3 résistances de 220 Ω ;
- un bouton-poussoir à montage en façade ;
- un bouton-poussoir tactile ;
- une pile de 9 V ;
- un connecteur pour la pile de 9 V ;
- du fil de câblage ;
- une bille ;
- une boîte en carton ;
- du papier épais ou cartonné ;
- de la colle blanche ;
- du ruban adhésif ou de la bande adhésive de masquage ;
- de la peinture ou des crayons-feutres ;
- des ciseaux ou un couteau de bricolage ;
- un fer à souder ;
- de la soudure.

Copyright © 2015 Eyrolles.

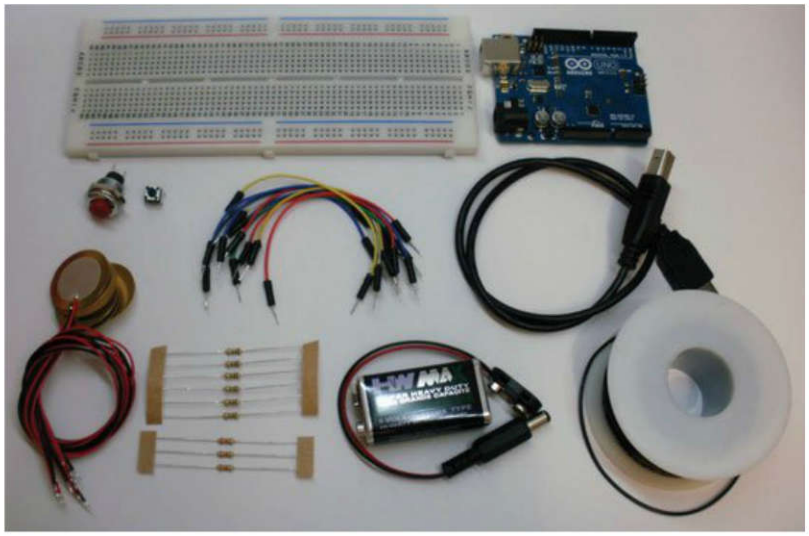


FIGURE 9-2 Les composants électroniques dont tu as besoin pour réaliser ton labyrinthe.

Première partie : le décompte des points

Dans l'aventure 5, tu as découvert les éléments piézoélectriques, que tu as utilisés comme actionneurs pour transformer une variation de tension en vibrations sonores. Mais ce n'est pas tout : ils sont aussi capables de convertir des vibrations en tensions électriques variables.

Dans cette aventure, tu vas utiliser des capteurs piézoélectriques pour détecter dans quel trou du labyrinthe la bille tombe. Chaque trou vaut un nombre de points différent. Ainsi, lors de la conception de ton labyrinthe, tu devras faire en sorte que les trous les plus difficiles à atteindre rapportent le plus de points.

Détection des vibrations avec des capteurs piézoélectriques

Les éléments piézoélectriques ont la faculté de produire des pointes de tension qui pourraient endommager ta carte Arduino. Tu dois donc utiliser une résistance de grande valeur afin de protéger ta carte Arduino. Une résistance de 100 MΩ est le plus souvent utilisée (cela fait 100 millions d'ohms).

Un capteur piézoélectrique produit une tension variable, tu ne dois pas détecter uniquement les niveaux de tension `HIGH` (HAUT) et `LOW` (BAS), mais également les niveaux intermédiaires. Tu vas donc devoir utiliser une broche analogique et utiliser la fonction `analogRead()` pour lire la tension.

Copyright © 2015 Eyrolles.

Construis le circuit de la figure 9-3 en suivant les étapes ci-dessous.

1. Relie le fil rouge du capteur piézoélectrique à une rangée courte au milieu de ta plaque de montage et son fil noir à l'une des grandes rangées en bas de ta plaque.
2. Connecte l'un des fils d'une résistance d'1 MΩ à la rangée où se situe le fil rouge et son autre fil à la rangée où se situe le fil noir.
3. Utilise un fil de pontage pour relier la rangée avec le fil rouge et la résistance à la broche A0 de l'Arduino Uno.
4. Utilise un autre fil de pontage pour relier la rangée avec le fil noir et l'autre résistance à la broche GND.

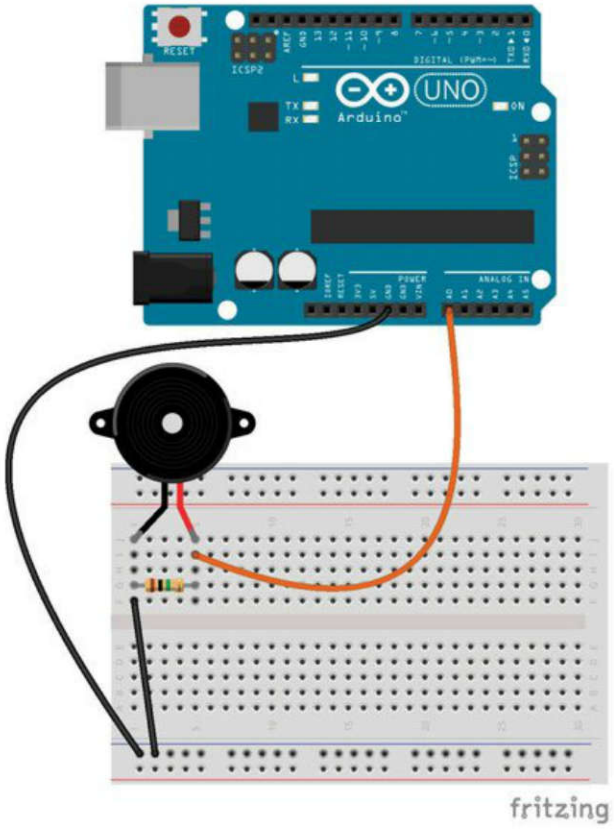


FIGURE 9-3 Le circuit pour l'utilisation d'un capteur piézoélectrique.

Démarre l'IDE Arduino, puis ouvre l'exemple à partir du menu Fichier > Exemples > 06.Sensor > Knock. Télécharge-le dans ta carte Arduino Uno et ouvre le moniteur série. Teste le sketch en tapant sur le capteur piézoélectrique. Observe le comportement de la LED reliée à la broche 13 et regarde ce qui est affiché sur le moniteur série. Tu devrais voir la LED s'allumer et s'éteindre quand tu tapes sur le capteur.

Copyright © 2015 Eyrolles.

Définir un seuil pour le marquage des points

Tu viens d'apprendre à lire une valeur à partir d'un capteur piézoélectrique, mais tu as besoin de plusieurs capteurs pour rendre ton jeu difficile ! Dans l'étape suivante, tu vas en ajouter quatre pour en avoir cinq au total.

Sur la même plaque de montage que celle où tu as un capteur relié à la broche A0, connecte quatre autres capteurs piézoélectriques de façon similaire. Relie-les aux sorties analogiques A1, A2, A3 et A4 comme sur la figure 9-4. Ignore, pour l'instant, le dernier élément piézoélectrique (celui qui est relié à la broche 9).

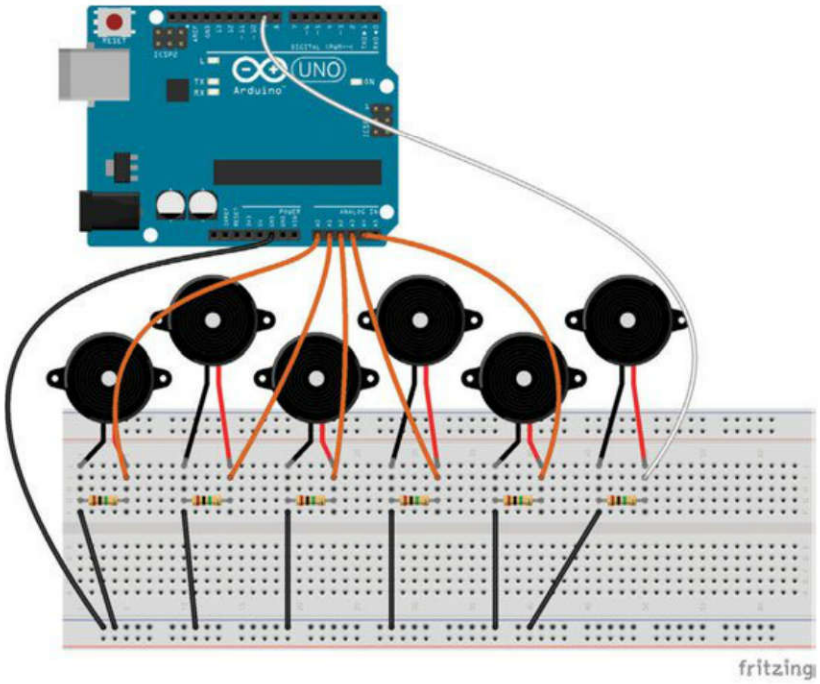


FIGURE 9-4 Le circuit pour cinq éléments piézoélectriques utilisés comme capteurs et un sixième utilisé comme haut-parleur.

Lance l'IDE Arduino et ouvre un nouveau sketch. Commence par créer les fonctions `setup()` et `loop()` vides.

```
void setup() {

}

void loop() {

}
```

Copyright © 2015 Eyrolles.

Comme tu dois gérer le fonctionnement de cinq capteurs piézoélectriques, il est préférable de les mémoriser dans un tableau. Au début du sketch, déclare un tableau qui mémorisera les numéros des broches des capteurs piézoélectriques et une variable qui contiendra le nombre de capteurs utilisés.

```
// Les variables pour les broches
int brochesPoints[] = {A0, A1, A2, A3, A4};

// Le nombre de capteurs pour le score
int nombrebrochesPoints = 5;
```

Tu n'as pas à fixer le mode des broches analogiques quand tu les utilises toutes en entrée. La seule ligne de code que tu dois ajouter à ta fonction `setup()` concerne donc l'initialisation de la communication série.

```
Serial.begin(9600);
```

Dans la fonction `loop()`, tu vas lire les valeurs transmises par chaque capteur piézoélectrique. Puisque les numéros des broches sont mémorisés dans un tableau, tu peux utiliser une boucle `for`. Pour l'instant, affiche simplement ces valeurs dans le moniteur série.

Ajoute les lignes de code suivantes dans `loop()`, télécharge ensuite le sketch dans ta carte Arduino Uno et ouvre le moniteur série de l'IDE Arduino :

```
int i;
for( i=0; i<nombrebrochesPoints; i++ ) {
  int valeurBroche = analogRead(brochesPoints[i]);
  // Affiche le numéro de la broche et la valeur lue
  Serial.print("Capteur : ");
  Serial.print(i);
  Serial.println(" Valeur : ");
  Serial.println(valeurBroche);
}
```

Tape successivement sur les capteurs et note les valeurs qui sont générées. Laisse ensuite tomber la bille que tu vas utiliser dans ton labyrinthe sur un des capteurs et note les valeurs obtenues.

Tu vas maintenant devoir modifier ton sketch afin qu'un message ne soit affiché que quand un capteur piézoélectrique transmettra une valeur supérieure à un seuil. En d'autres termes, un message ne sera affiché que quand les capteurs piézoélectriques seront soumis à une force suffisante.

Au début de ton sketch, entre les déclarations de variables et la fonction `setup()`, ajoute une variable qui définira le seuil à partir duquel un capteur piézoélectrique devra

Copyright © 2015 Eyrolles.

déclencher l'enregistrement du score. Définis le seuil en relation avec la valeur obtenue lors du test précédent avec ta bille :

```
int seuilPiezo = 800;
```

Dans la fonction `loop()`, ajoute une instruction `if` qui vérifiera si le capteur en cours de lecture a transmis une valeur supérieure au seuil défini. Quand celui-ci sera dépassé, le numéro du capteur et sa valeur seront affichés dans le moniteur série.

```
int i;
for( i=0; i<nombrebrochesPoints; i++ ) {
  int valeurBroche = analogRead(brochesPoints[i]);
  if( valeurBroche > seuilPiezo) {
    // Affiche le numéro de la broche et la valeur lue
    Serial.print("Capteur : ");
    Serial.print(i);
    Serial.println(" Valeur : ");
    Serial.println(valeurBroche);
    // Pause évitant à la bille de marquer le score
    // plusieurs fois
    delay(300) ;
  }
}
```

Télécharge ton sketch ainsi modifié et vérifie que chaque capteur piézoélectrique n'affiche un message que lorsque tu laisses ta bille tomber dessus. Voici ton sketch modifié tel qu'il se présente :

```
// Les variables pour les broches
int brochesPoints[] = {A0, A1, A2, A3, A4};

// Le nombre de capteurs pour le score
int nombrebrochesPoints = 5;

// Le seuil de déclenchement des capteurs piézoélectriques
int seuilPiezo = 800 ;

// setup() s'exécute une fois quand la carte est alimentée
// ou après avoir appuyé sur le bouton Reset
void setup() {
  Serial.begin(9600);
}

// loop() s'exécute de façon continue après la fonction setup()
void loop() {
  int i;
  for( i=0; i<nombrebrochesPoints; i++ ) {
```

Copyright © 2015 Eyrolles.

```
    int valeurBroche = analogRead(brochesPoints[i]);

    // Si la valeur ci-dessus est supérieure au seuil défini
    if( valeurBroche > seuilPiezo) {
      // Afficher le numéro de la broche et la valeur lue
      Serial.print("Capteur : ");
      Serial.print(i);
      Serial.println(" Valeur : ");
      Serial.println(valeurBroche);

      // Pause évitant à la bille de marquer le score
      // plusieurs fois
      delay(300) ;
    }
  }
}
```

Ajouter des effets sonores

Il est temps maintenant de voir ce que le sixième élément piézoélectrique est censé faire ! Dans la section précédente, tu as utilisé cinq éléments piézoélectriques en tant que capteurs ; le sixième élément piézoélectrique se comporte, lui, comme un haut-parleur. Sur la figure 9-4, le circuit du sixième capteur ressemble à celui des cinq autres, ce qui peut être source de confusion. La seule différence est qu'il est relié à la broche de sortie numérique 9 et non à une broche d'entrée analogique.

Tu vas utiliser la broche 9 afin d'appeler la fonction `tone()` pour produire un son avec cet élément piézoélectrique. Cependant, bien qu'il soit utilisé comme haut-parleur, cela ne l'empêche pas de pouvoir générer un pic de tension comme le ferait un capteur. Si la bille venait à heurter ton haut-parleur piézoélectrique, ce dernier pourrait produire un pic de tension susceptible d'endommager ta carte Arduino Uno. Tu dois donc encore utiliser un circuit de protection.

Continue en ajoutant le dernier élément piézoélectrique relié à la broche 9 comme sur la figure 9-3.

Dans le code de ton sketch, ajoute une variable avant la fonction `setup()` pour tenir compte du haut-parleur :

```
int brocheHautParleur = 9 ;
```

Immédiatement après avoir affiché les messages dans ta boucle `for`, juste avant l'instruction `delay()`, ajoute les lignes de code en caractères gras suivantes pour produire un son à chaque fois qu'un capteur piézoélectrique est déclenché au-delà du seuil fixé :

```
// Si la valeur ci-dessus est supérieure au seuil défini
if( valeurBroche > seuilPiezo) {
```

Copyright © 2015 Eyrolles.

```
  // Afficher le numéro de la broche et la valeur lue
  Serial.print("Capteur : ");
  Serial.print(i);
  Serial.println(" Valeur : ");
  Serial.println(valeurBroche);

  // Produire une tonalité sonore
  tone(brocheHautParleur, 659, 300) ;
  delay(300) ;

  // Pause évitant à la bille de marquer le score plusieurs
  // fois
  delay(300) ;
}
```

Télécharge ton sketch dans ta carte Arduino Uno et vérifie que tu entends bien un son à chaque fois qu'un capteur piézoélectrique est déclenché.

Mémoriser le score

Maintenant que tu disposes de cinq capteurs piézoélectriques pour détecter les chutes de la bille et d'un sixième pour reproduire un son, tu peux continuer en affectant les points à attribuer à chaque capteur et donc mémoriser ton score.

En premier, tu dois créer une variable pour mémoriser le score. Au début de ton sketch, à la suite des autres variables, ajoute la ligne suivante :

```
int scoreActuel = 0;
```

À l'intérieur de l'instruction `if`, dans la boucle `for` de la fonction `loop()`, tu dois faire trois modifications. En premier, crée une variable qui va retenir le nombre de points que tu viens de marquer et ajouter sa valeur au score. Ensuite, tu dois afficher le nombre de points qui viennent d'être ajoutés au score et le nouveau score ainsi obtenu dans le moniteur série :

```
// Si la valeur ci-dessus est supérieure au seuil
if( valeurBroche > seuilPiezo) {
  // ajout des points
  int nouveauxPoints = (i+1)*10 ;
  scoreActuel += nouveauxPoints ;

  // Afficher les points et le nouveau score
  Serial.print("Score ! ");
  Serial.print(nouveauxPoints);
  Serial.println(" points");
  Serial.print(scoreActuel);
  Serial.println(" points");
```

Copyright © 2015 Eyrolles.

```
  // Produire une tonalité sonore
  tone(brocheHautParleur, 659, 300) ;
  delay(300) ;

  // Pause évitant à la bille de marquer le score plusieurs
  // fois
  delay(300) ;
}
```

Le nombre de points est obtenu en ajoutant 1 au numéro d'index, dans le tableau, de la broche qui vient d'être déclenchée (la variable `i`) puis en le multipliant par 10. Par exemple, le capteur relié à la broche d'index 3 du tableau rapportera (3+1)*10, soit 40 points.

Le programme suivant constitue le sketch complet avec les modifications représentées en caractères gras. Télécharge-le dans ta carte Arduino Uno et ouvre le moniteur série. Vérifie que les effets sonores et les messages indiquant les points marqués apparaissent comme tu l'espères :

```
// Les variables pour les broches
int brochesPoints[] = {
A0, A1, A2, A3, A4};

// Le nombre de capteurs pour le score
int nombrebrochesPoints = 5;

// Le seuil de déclenchement des capteurs piézoélectriques
int seuilPiezo = 800 ;

// La broche du haut-parleur
int brocheHautParleur = 9;

// Le score total
int scoreActuel = 0;

// setup() s'exécute une fois quand la carte est alimentée
// ou après avoir appuyé sur le bouton Reset
void setup() {
  Serial.begin(9600);
}

// loop() s'exécute de façon continue après la fonction
// setup()
void loop() {
  int i;
  for( i=0; i<nombrebrochesPoints; i++ ) {
    int valeurBroche = analogRead(brochesPoints[i]);
```

Copyright © 2015 Eyrolles.

```
    // Si la valeur ci-dessus est supérieure au seuil
    if( valeurBroche > seuilPiezo) {
      // ajout des points
      int nouveauxPoints = (i+1)*10 ;
      scoreActuel += nouveauxPoints ;

      // Afficher les points et le nouveau score
      Serial.print("   Points marqués ! ");
      Serial.print(nouveauxPoints);
      Serial.println(" points");
      Serial.print("   Score actuel : ");
      Serial.print(scoreActuel);
      Serial.println(" points");

      // Produire une tonalité sonore
      tone(brocheHautParleur, 659, 300) ;
      delay(300) ;

      // Pause évitant à la bille de marquer le score
      // plusieurs fois
      delay(300) ;
    }
  }
}
```

Tu peux télécharger tous les sketches (en versions française ou anglaise) depuis le site d'accompagnement www.editions-eyrolles.com/go/arduino.

Deuxième partie : création du labyrinthe

Te voici enfin prêt à commencer la création de ton labyrinthe ! Tu dois tout d'abord trouver une boîte pour ton jeu. Sa taille exacte n'a pas d'importance. Elle doit être assez grande pour contenir tous les capteurs et ta carte Arduino Uno, mais il est préférable qu'elle ne le soit pas trop afin de pouvoir être tenue facilement entre tes mains. Une boîte d'environ 25 cm x 33 cm x 8 cm fera l'affaire.

Copyright © 2015 Eyrolles.

Si tu ne trouves qu'une boîte de longueur et de largeur suffisantes, mais trop haute, utilise des ciseaux ou un couteau de bricolage pour réduire sa hauteur. Tu peux te faire aider par un adulte, si c'est difficile à couper.

Dessiner le labyrinthe

Quand ta boîte est prête, fais un dessin du labyrinthe tel que tu le souhaites, tout en respectant les règles suivantes (voir la figure 9-5).

- Définis ton jeu selon cinq colonnes et décide, pour chacune, où tu vas situer les trous. Il n'y aura qu'un seul capteur piézoélectrique par colonne, tu ne peux donc avoir qu'un seul trou dans chacune. Définis ensuite le nombre de points que tu souhaites affecter à chaque trou : un premier vaudra 10 points, un autre 20, un autre 30, un autre 40 et le dernier, 50. Fais en sorte que le plus difficile à atteindre rapporte le plus de points et que le plus facile ne rapporte que le minimum.
- Il est important de laisser un espace vide en bas de ton labyrinthe. C'est là que ta bille roulera après avoir marqué des points. Tu dois laisser assez d'espace pour la récupérer facilement pour rejouer.

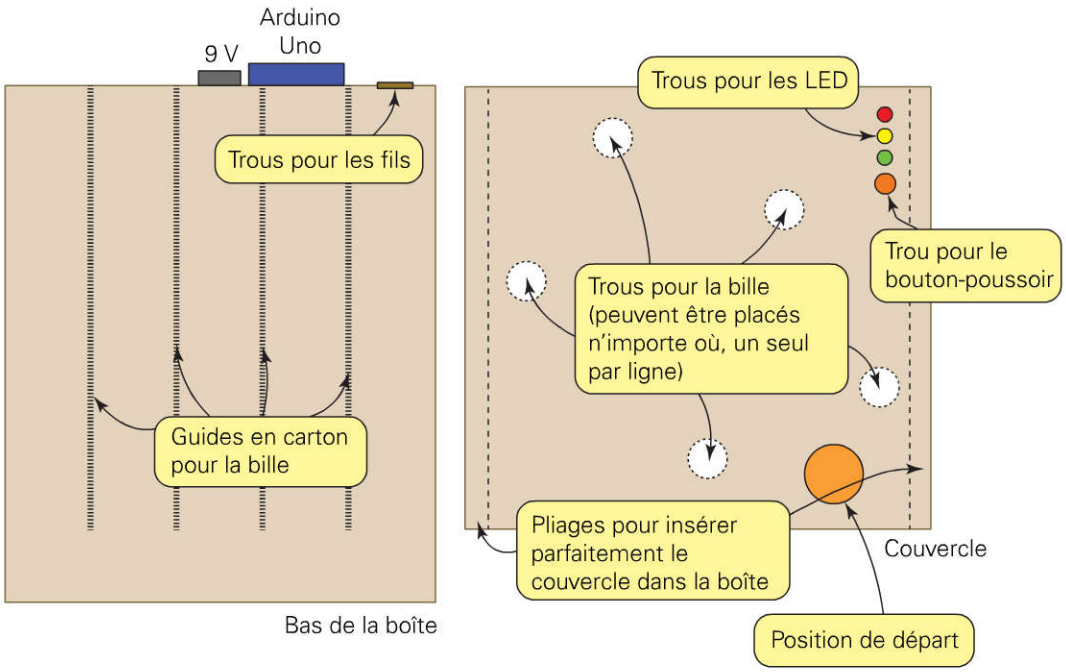


FIGURE 9-5 Les règles à respecter pour définir ton labyrinthe.

Copyright © 2015 Eyrolles.

- Réserve également une partie en haut à droite pour y mettre trois LED et un bouton-poussoir. Elles t'indiqueront quand commencer, quand le temps imparti s'est écoulé et quand tu auras atteint un nouveau score maximal. Réserve assez de place pour placer les trois LED et le bouton-poussoir ainsi que leurs étiquettes. Une surface d'environ 5 x 8 cm devrait convenir.
- Décide de l'endroit où tu devras placer la bille à chaque début de partie.

Maintenant que tu as défini la position des trous et des autres éléments, tu es prêt à concevoir ton labyrinthe ! Dessine son plan sur une feuille de papier et ne la perds pas, car tu vas en avoir besoin par la suite.

Créer les règles du jeu

Voici les règles du jeu du labyrinthe.

1. Le joueur appuie sur le bouton-poussoir de départ. La LED rouge s'illumine, puis la jaune, puis la verte. La musique de départ retentit ensuite et la partie commence : le joueur place sa bille dans la position de départ.
2. Le joueur essaie de déplacer la bille vers les différents trous pour marquer des points.
3. Une tonalité retentit quand des points sont marqués.
4. Chaque fois que le joueur marque des points, la bille roule en bas de la boîte. Le joueur replace alors la bille dans la position de départ et essaie à nouveau de marquer des points.
5. La partie continue ainsi jusqu'à ce que le temps imparti s'épuise.
6. Si le joueur a atteint un nouveau record, un son spécial retentit et la LED verte clignote. S'il n'a pas atteint le score maximal, un son différent retentit et la LED rouge clignote.
7. Si la carte Arduino Uno est réinitialisée ou éteinte, le score le plus élevé est effacé. Sinon, le joueur peut commencer une nouvelle partie pour tenter de battre son précédent record.

Ces différentes étapes peuvent être divisées en fonctions dans le programme, qui pourront être appelées au bon moment pendant le déroulement du jeu. La figure 9-6 montre comment elles interagissent dans le sketch Arduino que tu vas écrire par la suite.

Copyright © 2015 Eyrolles.

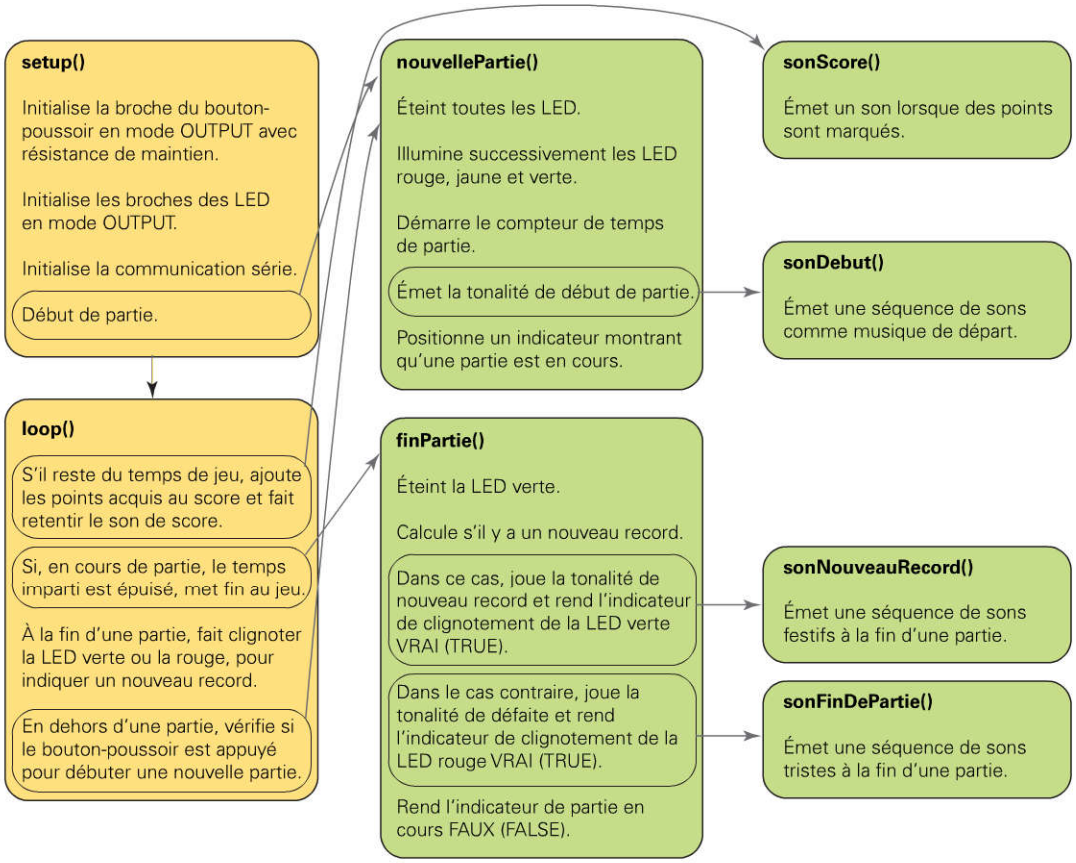


FIGURE 9-6 Le fonctionnement du programme lorsqu'une partie est jouée.

Réaliser le prototype du circuit

Tu as déjà réalisé la majeure partie du circuit du labyrinthe : tu as construit le circuit avec les cinq capteurs et le haut-parleur. Les composants qui manquent sont les trois LED et le bouton-poussoir. La figure 9-7 montre le schéma du circuit du labyrinthe complet.

Tu dois d'abord tester ton circuit sur une plaque de montage avant de construire ton labyrinthe, mais tu le sais bien maintenant ! Réalise le circuit de la figure 9-8.

1. Commence en reprenant le circuit que tu as déjà réalisé avec les éléments piézoélectriques.
2. Ajoute les trois LED sur la plaque de montage : une rouge, une jaune et une verte. Relie leurs fils négatifs à la grande rangée connectée à la masse GND.
3. Place une résistance de limitation de courant de 220 Ω avant chaque LED.
4. Utilise un fil de pontage pour relier la résistance placée avant la LED rouge à la broche 6, celle avant la LED jaune à la broche 5 et celle avant la LED verte à la broche 4.

Copyright © 2015 Eyrolles.

5. Insère le bouton-poussoir au-dessus du séparateur au milieu de ta plaque de montage afin de relier une des bornes à la grande rangée connectée à la masse GND. Relie l'autre borne à la broche 7.

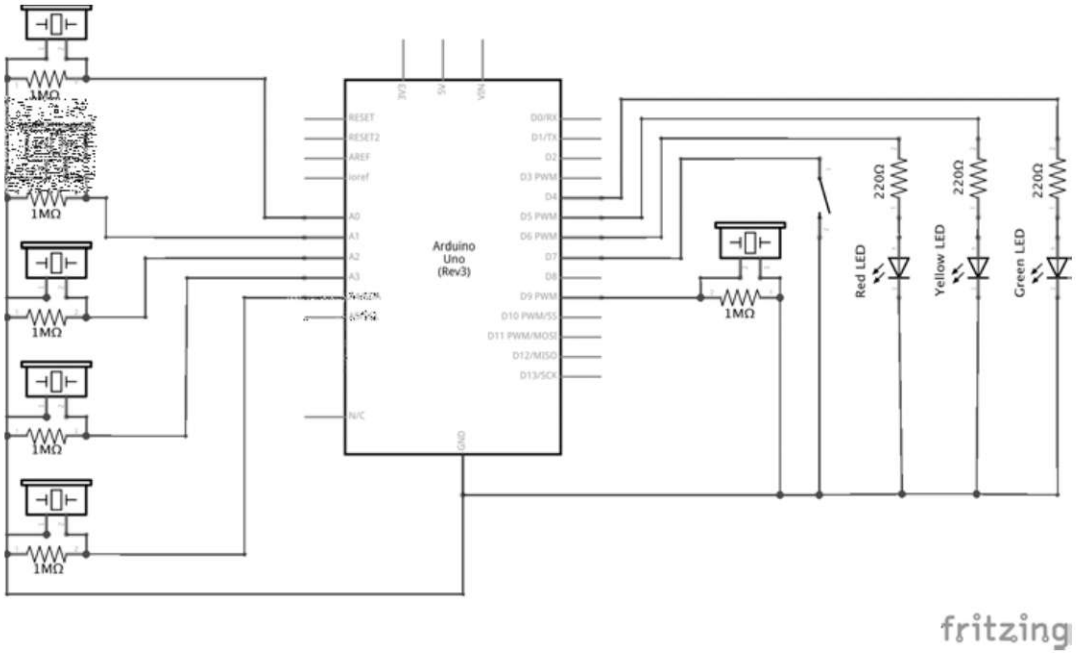


FIGURE 9-7 Le schéma du circuit du labyrinthe à bille.

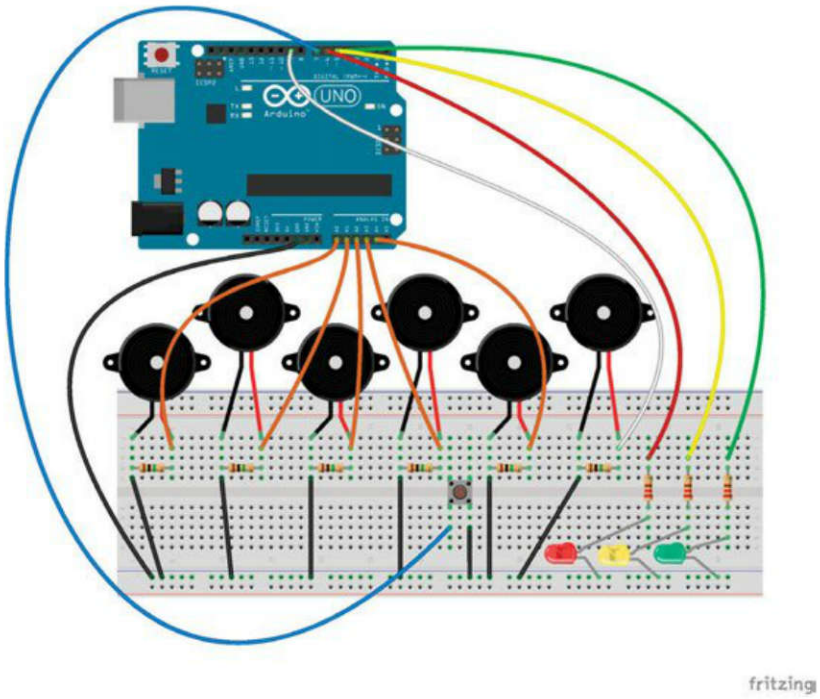


FIGURE 9-8 Le prototype du circuit du labyrinthe sur une plaque de montage.

Copyright © 2015 Eyrolles.

Troisième partie : écriture du programme

Te voici parvenu à l'écriture du sketch Arduino le plus long de toutes les aventures. Mais ne t'inquiète pas : tu es prêt ! Tu as déjà vu les différentes parties du code. Tu es sur le point de tout rassembler dans un sketch plus complexe, mais qui est simplement constitué de plusieurs petits bouts de code. Tu vas commencer par le programme de comptabilisation des points que tu as écrit précédemment dans cette aventure et y ajouter, petit à petit, de nouvelles fonctionnalités. Je vais te dire quand commencer.

Début de partie

Les LED doivent indiquer le compte à rebours du temps avant le commencement d'une nouvelle partie. Tu peux également afficher un message dans le moniteur série pour visualiser des informations additionnelles, si ton Arduino Uno est reliée à ton ordinateur. Les premières lignes de code sont les déclarations des variables pour les broches des LED et le bouton-poussoir. Au début de ton sketch, ajoute les lignes suivantes :

```
int LEDVerte = 4;
int LEDJaune = 5;
int LEDRouge = 6;
int brochePoussoir = 7;
```

Afin de bien organiser ton sketch, tu vas créer une fonction qui regroupera toutes les actions qui doivent être exécutées quand une nouvelle partie commence. Ajoute le code suivant à la fin de ton sketch, après la fonction `loop()` :

```
void nouvellePartie() {
  // S'assurer que les LED sont éteintes au départ
  digitalWrite(LEDRouge, LOW);
  digitalWrite(LEDJaune, LOW);
  digitalWrite(LEDVerte, LOW);

  Serial.println("****NOUVELLE PARTIE****");
  Serial.print("Début de la partie dans ._._.");

  // Allumer la LED rouge
  digitalWrite(LEDRouge, HIGH);
  Serial.print("À vos marques._._.");
  delay(1000);

  // Éteindre la LED rouge
  digitalWrite(LEDRouge, LOW);

  // Allumer la LED jaune
  digitalWrite(LEDJaune, HIGH);
  Serial.print("prêt._._.");
```

Copyright © 2015 Eyrolles.

```
  delay(1000);

  // Éteindre la LED jaune
  digitalWrite(LEDJaune, LOW);
  Serial.println("partez !");

  // Allumer la LED verte
  digitalWrite(LEDVerte, HIGH);

  // Remettre le score à zéro
  scoreActuel = 0;
}
```

La fonction `nouvellePartie()` s'assure que toutes les LED sont éteintes au début d'une nouvelle partie, puis elle les allume une par une. Elle affiche également les messages du compte à rebours. À la fin de la fonction, la variable `scoreActuel` mémorisant le score en cours est remise à zéro.

Dans la fonction `setup()`, définis le mode de fonctionnement des broches de chaque LED en sortie (OUTPUT) à l'aide de `pinMode()`. Appelle ensuite la fonction que tu viens d'écrire afin qu'une nouvelle partie démarre automatiquement lorsque ta carte Arduino Uno est mise en marche. Le nouveau code à ajouter est en caractères gras :

```
void setup() {
  // Définition du mode de la broche du bouton-poussoir
  pinMode(brochePoussoir, INPUT_PULLUP);

  // Définition du mode des broches des LED
  pinMode(LEDVerte, OUTPUT);
  pinMode(LEDJaune, OUTPUT);
  pinMode(LEDRouge, OUTPUT);

  // Initialisation de la communication série
  Serial.begin(9600);

  // Commencer le compte à rebours avant le démarrage
  nouvellePartie();
}
```

Télécharge ton sketch, ouvre le moniteur série et vérifie que tout fonctionne. Tes LED doivent s'allumer dans le bon ordre et les messages de compte à rebours doivent apparaître dans le moniteur série.

Fin de partie

Pour que ton labyrinthe devienne un jeu, tu dois ajouter un chronomètre afin que le joueur puisse marquer le maximum de points dans un temps donné.

Copyright © 2015 Eyrolles.

Pour créer un chronomètre dans ton code, commence par ajouter les variables suivantes au début de ton sketch. Tu ne vas pas encore toutes les utiliser, mais elles le seront quand ton jeu sera terminé :

```
int tempsMaximum = 10000;
long heureDeDepart;
boolean partieEnCours = false;
```

`heureDeDepart` est la première variable que tu vas utiliser. Tu as probablement remarqué qu'elle n'est pas de type `int` (entier), mais de type `long`. Tu as déjà rencontré ce type de données dans l'aventure 6. Il permet de stocker des valeurs plus élevées qu'un entier `int`, ce qui était nécessaire pour utiliser la bibliothèque des capteurs capacitifs lors de la programmation de ta boule de cristal. Dans le cas présent, la variable `heureDeDepart` mémorise le temps défini en millisecondes depuis le début de l'exécution du sketch dans ta carte Arduino Uno. Cela peut donc être un nombre très élevé, c'est pourquoi cette variable doit être de type `long` au lieu de `int`.

Afin de mémoriser l'heure à laquelle la partie commence, tu dois utiliser `millis()`, une fonction intégrée de l'IDE Arduino. Tu n'as donc pas à importer de bibliothèque pour pouvoir l'utiliser. Elle renvoie le temps écoulé depuis le départ du sketch. Tu vas mémoriser ce nombre afin de pouvoir vérifier ensuite combien de temps s'est écoulé depuis que la partie a débuté.

Ajoute les lignes de code suivantes à la fin de la fonction `nouvellePartie()` :

```
heureDeDepart = millis() ;
```

Dans la fonction `loop()`, tu dois maintenant vérifier si le temps écoulé depuis le début de la partie est épuisé. La variable `tempsMaximum` contient le temps qu'une partie doit durer, qui est fixé à 10 000 millisecondes (10 secondes), mais tu peux le réduire ou l'augmenter.

Ajoute l'instruction `if` suivante au début de la fonction `loop()`. L'ensemble du code que tu as écrit jusqu'à présent dans ta boucle pour lire les capteurs piézoélectriques et comptabiliser le score doit maintenant être situé à l'intérieur des crochets de ce `if`. De cette façon, les nouveaux points ne seront additionnés que si le temps de jeu n'est pas épuisé :

```
if( (millis() - heureDeDepart) < tempsMaximum) {
// code déjà écrit pour comptabiliser les points marqués
}
```

Cette instruction `if` vérifie que le temps écoulé depuis le début de la partie est inférieur à la valeur `tempsMaximum`.

`partieEnCours` fait partie des variables que tu as ajoutées au début de ton sketch. Elle est de type booléen : cela signifie qu'elle peut uniquement être égale à `true` (vrai)

Copyright © 2015 Eyrolles.

ou `false` (faux). Tu utilises cette variable comme un drapeau ou un fanion. Chaque fois qu'une partie commence, elle est mise à la valeur `true`, et quand aucune partie n'est en cours, elle est mise à la valeur `false`. Tu peux alors prendre des décisions dans ton code selon qu'une partie est en cours ou non.

> Un **drapeau** ou **fanion** (*flag* en anglais) est une variable dans un programme qui mémorise l'état d'une autre partie du code. C'est en général une variable booléenne.
>
>
>

À la fin de la fonction `nouvellePartie()`, ajoute cette instruction qui met le drapeau `partieEnCours` à la valeur `true` (vrai).

```
partieEnCours = true ;
```

Crée maintenant une nouvelle fonction nommée `finPartie()` à la fin de ton sketch. Celle-ci sera appelée quand le temps de partie sera écoulé entièrement. Elle mettra alors le drapeau `partieEnCours` à la valeur `false` (faux) :

```
void finPartie() {
  Serial.println("Partie terminée !");
  Serial.print("Score : ");
  Serial.println(scoreActuel);

  // Positionne le drapeau partieEnCours pour indiquer
  // qu'il n'y a plus de partie en cours
  partieEnCours = false;
}
```

Dans la boucle `loop()`, la première instruction `if` vérifie si la partie en cours a lieu dans le temps imparti. Si ce n'est pas le cas, tu dois cesser de comptabiliser les points qui sont marqués. Si la partie est en cours et que le temps imparti est entièrement écoulé, tu dois mettre fin à la partie. Enfin, si le temps limite est dépassé et qu'aucune partie n'est en cours, tu dois indiquer si un nouveau record a été battu ou non. Tu vas programmer la partie concernant un nouveau record dans la section suivante. Pour le moment, concentre-toi sur la fin de la partie.

Dans le code qui est exécuté uniquement si aucune partie n'est en cours ou si le temps limite est écoulé, tu dois utiliser une instruction `else`. Celle-ci contient le code qui doit être exécuté uniquement si la condition de l'instruction `if` est fausse. Ce code ne sera pas exécuté dans d'autres conditions, il doit donc être pairé avec un `if`.

Copyright © 2015 Eyrolles.

À l'intérieur de la fonction `loop()`, après avoir fermé l'accolade } de l'instruction `if`, ajoute les lignes suivantes :

```
else{
  // sinon, si une partie est en cours
  // et le temps limite est atteint
  if( partieEnCours) {
    // terminer la partie
    finPartie();
  }
}
```

Le bloc de code de l'instruction `else` est exécuté uniquement si le temps limite est écoulé. L'instruction `if` à l'intérieur de `else` vérifie si le drapeau `partieEnCours` est vrai. Si c'est le cas, la partie est alors terminée par l'appel à la fonction `finPartie()`.

Télécharge le sketch dans ta carte et vérifie que cela fonctionne. Cela devrait faire tout ce qui a été programmé jusqu'à présent : commencer une partie et tenir le score à jour. Maintenant, le programme devrait aussi terminer la partie après un temps maximum `tempsMaximum` fixé à 10 secondes.

Début d'une nouvelle partie

Maintenant que ton jeu se termine quand le temps limite est atteint, tu dois pouvoir commencer une nouvelle partie pour essayer de battre ton score ! Dans l'étape suivante, tu vas ajouter un bouton-poussoir qui permettra d'entamer une nouvelle partie. Ce bouton fait déjà partie du circuit sur ta plaque de montage, tu n'as donc plus qu'à ajouter le code correspondant.

Dans le bloc d'instructions `else` que tu as ajouté à l'étape précédente, ajoute un `else` associé au bloc `if` qui vérifie si le drapeau `partieEnCours` est vrai :

```
else{
  int valeurPoussoir = digitalRead( brochePoussoir );
  if( valeurPoussoir == 0) {
    // Le bouton est appuyé, commencer une nouvelle partie
    nouvellePartie();
  }
}
```

Maintenant, si le drapeau `partieEnCours` est faux, le bouton-poussoir est testé pour voir s'il est appuyé. Si c'est le cas, une nouvelle partie est commencée en appelant la fonction `nouvellePartie()`.

La fonction `loop()` contient maintenant le code suivant. Télécharge le sketch dans ta carte et vérifie que tout fonctionne. Une nouvelle partie doit commencer par un compte à rebours qui est indiqué par les LED quand tu démarres ta carte Arduino Uno pour la

Copyright © 2015 Eyrolles.

première fois, et elle doit se terminer au bout de 10 secondes. Si tu appuies sur le bouton-poussoir, une nouvelle partie commence :

```
void loop() {
  int i;
  if( (millis() - heureDeDepart) < tempsMaximum ){
    for( i=0; i<nombreBrochesPoints; i++ ) {
      int valeurBroche = analogRead(brochesPoints[i]);

      // Si la valeur ci-dessus est supérieure au seuil
      if( valeurBroche > seuilPiezo) {
        // Ajout des points
        int nouveauxPoints = (i+1)*10 ;
        scoreActuel += nouveauxPoints ;

        // Afficher le numéro de la broche et la valeur lue
        Serial.print("Capteur : ");
        Serial.print(i);
        Serial.println(" Valeur : ");
        Serial.println(valeurBroche);

        // Produire une tonalité sonore
        tone(brocheHautParleur, 659, 300) ;
        delay(300) ;

        // Pause évitant à la bille de marquer le score
        // plusieurs fois
        delay(300) ;
      }
    }
  }
  else{
    // else si une partie est en cours
    // et que le temps limite n'est pas atteint
    if( partieEnCours) {
      // terminer la partie
      finPartie();
    }
    else{
      int valeurPoussoir = digitalRead( brochePoussoir );
      if( valeurPoussoir == 0) {
        // Le bouton est appuyé, débuter une nouvelle partie
        nouvellePartie();
      }
    }
  }
}
```

Copyright © 2015 Eyrolles.

Cette fonction `loop()` devient compliquée ! La figure 9-9 montre globalement comment elle fonctionne, et ce que tu vas ajouter maintenant pour mémoriser le score maximal.

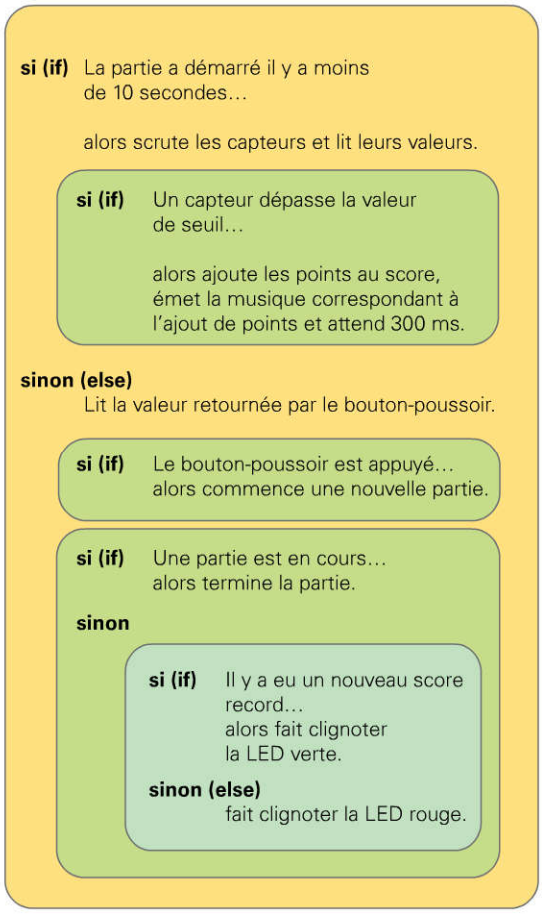


FIGURE 9-9 Le fonctionnement de la fonction loop().

Mémoriser le score maximal

Au tout début de ton sketch, ajoute trois nouvelles variables :

```
boolean nouveauScoreMaximum = false;
int scoreActuel;
int scoreMaximum = 0;
```

Tu viens d'ajouter un nouveau drapeau `nouveauScoreMaximum` qui indiquera si la dernière partie a atteint un nouveau record. Si c'est le cas, la LED verte clignotera quand aucune partie ne sera en cours. Sinon, c'est la LED rouge qui clignotera.

Dans la fonction `finPartie()`, ajoute le code suivant pour comparer le dernier score avec le score le plus élevé mémorisé. Si c'est un nouveau record, le drapeau devient `true` (vrai), sinon il est `false` (faux) :

Copyright © 2015 Eyrolles.

```
if( scoreActuel > scoreMaximum ) {
  // si c'est un nouveau record
  scoreMaximum = scoreActuel;
  Serial.println("Nouveau score record !");
  nouveauScoreMaximum = true;
}
else {
  // else pas de nouveau record
  nouveauScoreMaximum = false;
}
```

Tu dois maintenant ajouter le code qui fait clignoter les LED. Dans la fonction `loop()`, à l'intérieur du bloc `else` où tu vérifies le bouton-poussoir, ajoute le code suivant :

```
if( nouveauScoreMaximum ) {
  digitalWrite(LEDVerte, HIGH);
  delay(200);
  digitalWrite(LEDVerte, LOW);
  delay(200);
}
// clignotement de la LED rouge si le drapeau est faux
else {
  digitalWrite(LEDRouge, HIGH);
  delay(200);
  digitalWrite(LEDRouge, LOW);
  delay(200);
}
```

Télécharge le sketch et vérifie qu'il fonctionne. Tu dois être capable de commencer une partie quand l'Arduino Uno est mise sous tension ou remise à zéro (Reset). Quand la partie se termine, si tu as atteint un nouveau record, la LED verte doit clignoter, sinon, c'est la LED rouge qui doit clignoter jusqu'à ce que tu commences une nouvelle partie.

Si tu as des difficultés pour compiler ton programme en raison d'erreurs typographiques, tu peux télécharger le programme depuis le site d'accompagnement (www.editions-eyrolles.com/go/arduino). Parcours tout de même le code complet en le lisant afin de comprendre comment il fonctionne.

Ajouter les sons

Tu as déjà écrit le code qui émet des sons quand tu marques des points dans la première partie de cette aventure. Il ne te reste plus qu'à le placer dans sa propre fonction. De cette façon, ton programme sera plus facile à lire et il sera plus aisé de comprendre ce qui s'y passe.

Copyright © 2015 Eyrolles.

Déplace les deux lignes de code qui émettent un son quand un capteur piézoélectrique est déclenché de la fonction `loop()` vers sa propre fonction appelée `musiqueScore()`, que tu inséreras à la fin du sketch :

```
void musiqueScore() {
  tone (brocheHautParleur, 659, 300) ;
  delay(300) ;
}
```

Tu dois maintenant appeler cette nouvelle fonction dans `loop()`, juste après avoir affiché le score actuel :

```
musicScore() ;
```

Il ne te reste plus que trois séquences sonores à programmer. La première correspond à la musique qui sera émise quand une nouvelle partie commence. À la fin de ton sketch, ajoute une nouvelle fonction appelée `musiqueNouvellePartie()` comprenant le code suivant :

```
void musiqueNouvellePartie() {
  tone(brocheHautParleur, 523, 300);
  delay(300);
  tone(brocheHautParleur, 659, 300);
  delay(300);
  tone(brocheHautParleur, 784, 300);
  delay(300);
  tone(brocheHautParleur, 1047, 500);
  delay(600);
}
```

L'une des deux séquences sonores suivantes est émise à la fin d'une partie. Celle qui est émise dépend de l'atteinte ou non d'un nouveau record. La séquence correspondant à un nouveau record résonne plus gaiement.

À la fin de ton sketch, ajoute les fonctions `musiqueNouveauRecord()` et `musiqueFinTriste()` :

```
void musiqueNouveauRecord() {
  tone(brocheHautParleur, 880, 300);
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
```

Copyright © 2015 Eyrolles.

```
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
  delay(500);
}

void musiqueFinTriste() {
  tone(brocheHautParleur, 698, 300);
  delay(300);
  tone(brocheHautParleur, 622, 300);
  delay(300);
  tone(brocheHautParleur, 587, 300);
  delay(300);
  tone(brocheHautParleur, 523, 500);
  delay(600);
}
```

Ces deux fonctions sont appelées à l'intérieur de la fonction `FinDePartie()`. `musiqueNouveauRecord()` est appelée dans le bloc d'instructions `if` qui vérifie si un nouveau record a été atteint, et `musiqueFinTriste()` est appelée dans le bloc `else` qui suit :

```
void finPartie() {
  Serial.println("Partie terminée !");
  Serial.print("Score : ");
  Serial.println(scoreActuel);

  // Positionne le drapeau partieEnCours pour indiquer
  // qu'il n'y a plus de partie en cours
  partieEnCours = false;
  if( scoreActuel > scoreMaximum ) {
    // si c'est un nouveau record
    scoreMaximum = scoreActuel;
    Serial.println("Nouveau score record !");
    nouveauScoreMaximum = true;

    // émettre la musique pour un record
    musiqueNouveauRecord() ;
  }
  else{
    // sinon, si ce n'est pas un nouveau record
    nouveauScoreMaximum = false;
    // émettre la musique de fin de partie
    musiqueFinTriste() ;
  }
}
```

Copyright © 2015 Eyrolles.

C'est fini ! Tu as maintenant le sketch complet. Le programme entier figure ci-après, tu peux le comparer avec ce que tu as écrit. Télécharge le sketch dans ta carte Arduino Uno et teste-le avec ton circuit prototype réalisé sur ta plaque de montage. Quand tu seras satisfait de voir ton nouveau jeu fonctionner correctement, tu seras prêt pour terminer la réalisation de ton jeu du labyrinthe à bille !

DÉFI

Essaie de modifier les séquences d'appel de la fonction `tone()` qui correspondent au début et à la fin de partie de façon à personnaliser ton jeu !

```
// Les variables pour les broches
int brochesPoints[] = {A0, A1, A2, A3, A4};
int brochePoussoir = 7;
int LEDVerte = 4;
int LEDJaune = 5;
int LEDRouge = 6;

// La broche du haut-parleur
int brocheHautParleur = 9;

// Le nombre de capteurs pour le score
int nombrebrochesPoints = 5;

// Le seuil de déclenchement des capteurs piézoélectriques
int seuilPiezo = 800 ;

// Les variables des timers du jeu
int tempsMaximum = 10 000;
long heureDeDepart;
boolean partieEnCours = false;

// Les variables de score
boolean nouveauScoreMaximum = false;
int scoreActuel = 0;
int scoreMaximum = 0;

// setup() s'exécute une fois quand la carte est alimentée
// ou après avoir appuyé sur le bouton Reset
void setup() {
```

Copyright © 2015 Eyrolles.

```
  // Définition du mode de la broche du poussoir
  pinMode(brochePoussoir, INPUT_PULLUP);

  // Définition du mode des broches des LED
  pinMode(LEDVerte, OUTPUT);
  pinMode(LEDJaune, OUTPUT);
  pinMode(LEDRouge, OUTPUT);

  // Initialisation de la communication série
  Serial.begin(9600);

  // Commencer le compte à rebours avant le démarrage
  nouvellePartie();
}

// loop() s'exécute de façon continue après la fonction
// setup()
void loop() {
  // Si la partie est dans le temps imparti
  if( (millis() - heureDeDepart) < tempsMaximum) {
    // Lire les points de chaque broche
    int i;
    for( i=0; i<nombrebrochesPoints; i++ ) {
      int valeurBroche = analogRead(brochesPoints[i]);

      // Si la valeur ci-dessus est supérieure au seuil
      if( valeurBroche > seuilPiezo) {
        // Ajouter des points
        int nouveauxPoints = (i+1)*10 ;
        scoreActuel += nouveauxPoints ;

        // Afficher les points et le nouveau score
        Serial.print("   Points marqués ! ");
        Serial.print(nouveauxPoints);
        Serial.println(" points");
        Serial.print("   Score actuel : ");
        Serial.print(scoreActuel);
        Serial.println(" points");


        // Produire la tonalité sonore de marquage de points
        musiqueScore();

        // Pause évitant à la bille de marquer le score
        // plusieurs fois
        delay(300) ;
      }
    }
```

Copyright © 2015 Eyrolles.

```
  }
  else{
    // sinon, si une partie est en cours
    // et que le temps limite est atteint
    if( partieEnCours) {
      // terminer la partie
      finPartie();
    }
    else{
      // sinon, vérifier si le poussoir est appuyé pour
      // débuter une partie
      int valeurPoussoir = digitalRead( brochePoussoir );
      if( valeurPoussoir == 0) {
        // Le bouton est appuyé, débuter une nouvelle partie
        nouvellePartie();
      }
      // Clignotement de la LED verte en cas de nouveau
      // record
      if( nouveauScoreMaximum ) {
        digitalWrite(LEDVerte, HIGH);
        delay(200);
        digitalWrite(LEDVerte, LOW);
        delay(200);
      }
      // Clignotement de la LED rouge si le drapeau est faux
      else {
        digitalWrite(LEDRouge, HIGH);
        delay(200);
        digitalWrite(LEDRouge, LOW);
        delay(200);
      }
    }
  }
}

// Nouvelle partie :
// Définition des variables pour une nouvelle partie
// et du compte à rebours
void nouvellePartie() {
  // Assurer que les LED sont éteintes au départ
  digitalWrite(LEDRouge, LOW);
  digitalWrite(LEDJaune, LOW);
  digitalWrite(LEDVerte, LOW);

  Serial.println("****NOUVELLE PARTIE****");
  Serial.print("Début de la partie dans._._.");
```

Copyright © 2015 Eyrolles.

```
  // Allumer la LED rouge
  digitalWrite(LEDRouge, HIGH);
  Serial.print("À vos marques._._.");
  delay(1000);

  // Éteindre la LED rouge
  digitalWrite(LEDRouge, LOW);

  // Allumer la LED jaune
  digitalWrite(LEDJaune, HIGH);
  Serial.print("prêt._._.");
  delay(1000);

  // Éteindre la LED jaune
  digitalWrite(LEDJaune, LOW);
  Serial.println("partez !");

  // Allumer la LED verte
  digitalWrite(LEDVerte, HIGH);

  // Émettre la mélodie de début de partie
  musiqueNouvellePartie();

  // Démarrer le timer de partie
  heureDeDepart = millis() ;

  // Drapeau de partie à vrai
  partieEnCours = true ;

  // Remise à zéro du score
  scoreActuel = 0;
}

// Fin de partie :
// Affichage des messages de fin de partie
// et vérification du score par rapport au score record
void finPartie() {
  Serial.println("Partie terminée !");
  Serial.print("Score : ");

  Serial.println(scoreActuel);
  // Positionner le drapeau partieEnCours pour indiquer
  // qu'il n'y a plus de partie en cours
  partieEnCours = false;
  if( scoreActuel > scoreMaximum ) {
    // Si c'est un nouveau record
    scoreMaximum = scoreActuel;
    Serial.println("Nouveau score record !");
```

Copyright © 2015 Eyrolles.

```
    nouveauScoreMaximum = true;

    // Émettre la musique pour un record
    musiqueNouveauRecord() ;
  }
  else{
    // Sinon, si ce n'est pas un nouveau record
    nouveauScoreMaximum = false;
    // Émettre la musique de fin
    musiqueFinTriste() ;
  }
}

// Émettre la séquence sonore de marquage de points
void musiqueScore() {
  tone (brocheHautParleur, 659, 300) ;
  delay(300) ;
}

// Émettre la séquence sonore de début de partie
void musiqueNouvellePartie() {
  tone(brocheHautParleur, 523, 300);
  delay(300);
  tone(brocheHautParleur, 659, 300);
  delay(300);
  tone(brocheHautParleur, 784, 300);
  delay(300);
  tone(brocheHautParleur, 1047, 500);
  delay(600);
}

// Émettre la séquence sonore lors d'un nouveau score record
void musiqueNouveauRecord() {
  tone(brocheHautParleur, 880, 300);
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
  delay(200);
  tone(brocheHautParleur, 440, 500);
  delay(200);
  tone(brocheHautParleur, 880, 300);
  delay(500);
}
```

Copyright © 2015 Eyrolles.

```
// Émettre la séquence sonore de fin de partie
void musiqueFinTriste() {
  tone(brocheHautParleur, 698, 300);
  delay(300);
  tone(brocheHautParleur, 622, 300);
  delay(300);
  tone(brocheHautParleur, 587, 300);
  delay(300);
  tone(brocheHautParleur, 523, 500);
  delay(600);
}
```

Quatrième partie : fabrication du labyrinthe à bille

Maintenant que ton programme et ton circuit sont testés et qu'ils fonctionnent correctement, tu peux terminer la réalisation de ton labyrinthe. Tu vas utiliser une boîte sans couvercle, c'est-à-dire avec quatre côtés et le fond, mais pas de dessus. Tu fabriqueras un couvercle partiel pour la partie qui contiendra ton labyrinthe et les trous dans lesquels la bille tombera. Si tu veux décorer ta boîte avec de la peinture ou du papier, ce sera plus facile de le faire avant de découper les trous et d'assembler le circuit à l'intérieur.

Tu trouveras une vidéo (en anglais) montrant comment réaliser le labyrinthe sur le site d'accompagnement www.editions-eyrolles.com/go/arduino.

Fabriquer ton labyrinthe

Les étapes suivantes te permettront de créer ton labyrinthe.

1. Découpe quatre bandes de papier très épais ou cartonné de la longueur de ton labyrinthe et d'environ 8 cm de haut. Replie-les sur environ 1 cm afin de pouvoir les coller sur le fond de la boîte. Ces bandes empêcheront la bille de rouler au-dessus d'un capteur piézoélectrique après être tombée dans un trou, marquant ainsi, accidentellement, davantage de points. Colle les bandes de papier comme le montre la figure 9-10.
2. Découpe un autre morceau de papier cartonné d'environ 2,5 cm plus grand que la largeur du fond de ta boîte et d'environ 2,5 cm plus court que sa longueur. Tu peux replier les côtés qui dépassent sur environ 1,25 cm sous forme de volets qui maintiendront le labyrinthe fermement en place dans ta boîte. Regarde la figure 9-11 pour te guider.

Copyright © 2015 Eyrolles.

3. Reprends maintenant le morceau de papier sur lequel tu as dessiné ton labyrinthe au début de cette aventure. En l'utilisant comme guide, marque les emplacements des trous sur la plaque de papier cartonné que tu viens de découper. Repère également les trous pour les LED et le bouton-poussoir.

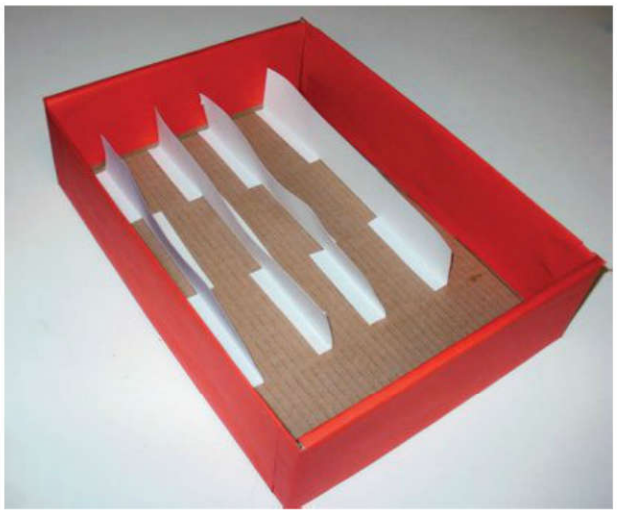


FIGURE 9-10 Colle les bandes de papier cartonné pour quider la bille après sa chute dans un trou.

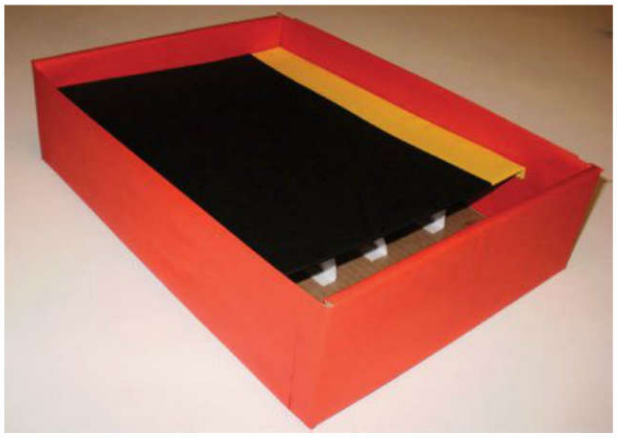


FIGURE 9-11 Le couvercle du labyrinthe placé et ajusté au fond de la boîte.

4. Perce les trous de la bille ainsi que ceux des LED et du bouton-poussoir dans la plaque cartonnée, comme sur la figure 9-12.
5. Crée des « murs » en papier cartonné afin de délimiter les passages de ton labyrinthe. Pour cela, découpe des bandes de papier cartonné que tu colleras le long des couloirs du labyrinthe.

Copyright © 2015 Eyrolles.

6. Colorie la plaque avec de la peinture ou des feutres, comme tu le souhaites. C'est là que tu peux donner libre cours à ton imagination ! Tu voudras probablement aussi indiquer le nombre de points que rapporte chaque trou.

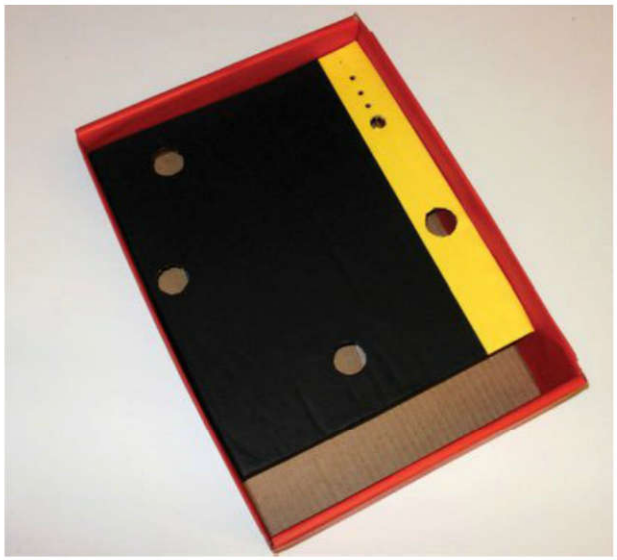


FIGURE 9-12 Le jeu du labyrinthe sans ses circuits électroniques.

Assembler les composants piézoélectriques

Tu vas maintenant assembler ton circuit. Suis les étapes ci-dessous pour placer les éléments piézoélectriques.

1. Repère les emplacements où tes capteurs piézoélectriques, le haut-parleur, les LED et le bouton-poussoir seront placés.
2. Dans le coin supérieur droit de ta boîte, perce un trou dans le carton afin de permettre aux fils qui seront à l'intérieur de la boîte de le traverser pour rejoindre la carte Arduino Uno à l'extérieur.
3. Coupe des fils qui relieront chaque fil rouge des éléments piézoélectriques à ta carte Arduino Uno. Découpe une autre série de fils afin de relier ensemble les fils noirs des éléments piézoélectriques.
4. Soude une résistance d'1 MΩ entre les fils rouge et noir des éléments piézoélectriques. Soude les fils prévus pour relier la carte Arduino aux fils des résistances qui sont connectés aux fils rouges des composants piézoélectriques. Soude les autres fils reliant les fils noirs des éléments piézoélectriques. Regarde la figure 9-13 pour te guider. Repère les fils par un marquage indiquant les broches auxquelles ils doivent être reliés.

Copyright © 2015 Eyrolles.

Ne commence à souder qu'en présence d'un adulte. Les opérations de soudage peuvent être dangereuses, alors sois prudent !

5. Insère les six fils soudés aux fils rouges des composants piézoélectriques dans leurs broches respectives sur ta carte Arduino Uno. Insère le fil soudé aux fils noirs dans la broche de masse GND. Télécharge maintenant le sketch du jeu et vérifie que tous les capteurs piézoélectriques fonctionnent. Tu devrais être capable de marquer des points avec les cinq capteurs, et le sixième élément piézoélectrique devrait émettre les séquences musicales et les effets sonores.

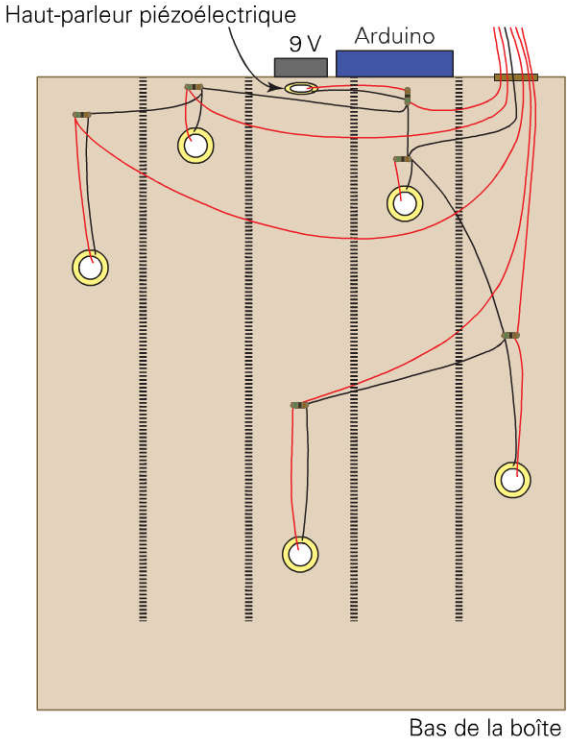


FIGURE 9-13 Le plan du câblage des composants piézoélectriques.

Assembler les LED et le bouton-poussoir

Lorsque tu es certain que la peinture et la colle que tu as utilisées pour décorer ton jeu sont sèches, suis les étapes ci-après pour placer les LED et le bouton-poussoir.

1. Soude une résistance de limitation de courant sur le fil positif de chaque LED. Découpe ensuite et soude les fils qui relient chaque résistance à ta carte Arduino Uno.

Copyright © 2015 Eyrolles.

2. Découpe et soude deux morceaux de fil qui relieront chaque broche du bouton-poussoir à ta carte Arduino.

3. Place les LED et le bouton-poussoir dans la plaque cartonnée formant le plateau de ton jeu. Soude le fil négatif des LED et une des broches du bouton-poussoir ensemble. La figure 9-14 te montre ce que tu devrais obtenir, une fois que tu auras terminé les soudures.

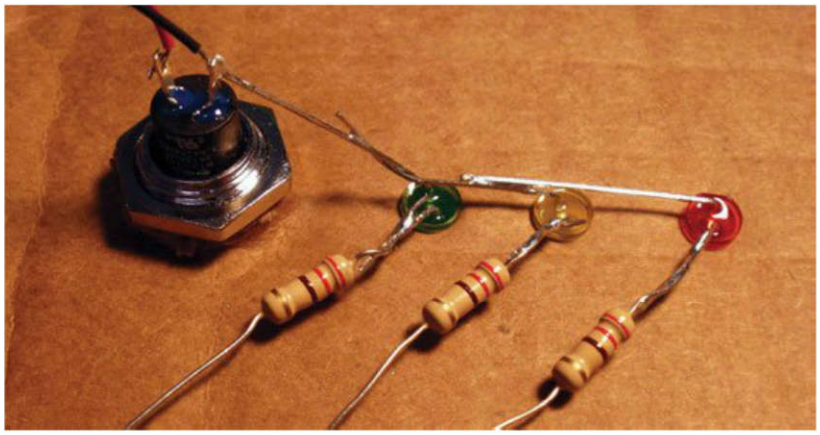


FIGURE 9-14 Soude ensemble les fils négatifs des LED et un contact du bouton-poussoir.

La touche finale

Tu as juste besoin d'une petite touche finale pour terminer cette aventure ! Suis les étapes ci-après pour assembler le tout.

1. Passe tous les fils provenant des composants piézoélectriques, des LED et du bouton-poussoir au travers du trou que tu as percé au fond de ta boîte. Attache ta carte Arduino Uno à l'extérieur de ta boîte près de ce trou avec du ruban adhésif.

2. Relie les fils à ta carte Arduino Uno, en connectant les capteurs piézoélectriques aux broches analogiques, et le haut-parleur, les LED et le bouton-poussoir aux broches numériques. Le fil qui est relié à tous les fils noirs des éléments piézoélectriques doit être inséré dans une broche de masse GND, et celui qui provient des fils négatifs des LED et du bouton-poussoir dans une autre broche de masse GND.

3. Vérifie que tes LED, le bouton-poussoir et les éléments piézoélectriques se comportent comme tu l'espères en jouant une partie de ton nouveau jeu du labyrinthe à bille.

C'est terminé ! Félicitations, tu es arrivé au bout de cette grande aventure ! C'était beaucoup de travail, mais j'espère que tu as trouvé cela enrichissant. Tu peux maintenant te détendre et jouer à ton nouveau jeu !

Copyright © 2015 Eyrolles.

Rappel des commandes Arduino de cette aventure	
Commande	**Description**
`boolean`	Type de donnée d'une variable qui peut prendre les valeurs `true` (VRAI) ou `false` (FAUX). Voir également `http://arduino.cc/en/Reference/BooleanVariables`.
`else`	Délimite un bloc d'instructions qui sont exécutées uniquement si l'instruction `if` qui précède n'est pas vérifiée (`false`). Voir également `http://arduino.cc/en/Reference/Else`.
`millis()`	Fonction retournant le temps écoulé depuis le début de l'exécution du sketch, en millisecondes. Voir également `http://arduino.cc/en/Reference/Millis`.

Encore plus d'aventures : va plus loin avec Arduino

Les microcontrôleurs Arduino sont un excellent moyen d'apprendre l'électronique et la programmation, mais ce n'est bien sûr que la moitié du plaisir. Comme tu as pu le voir, les projets ne prennent réellement vie que quand tu incorpores ton Arduino dans un objet. Tu as déjà manipulé beaucoup d'outils dans ce livre, alors pourquoi ne pas poursuivre en étudiant des techniques numériques comme la découpe laser ou l'impression 3D ? Pour débuter, lis par exemple le livre *Pratique de l'impression 3D* d'Anna Kaziunas France *et al.* (éditions Eyrolles, 2014). Tu peux aussi mettre à profit tes compétences acquises en électronique et en programmation pour utiliser les contrôleurs Raspberry Pi ! Lis par exemple le livre *À la découverte du Raspberry Pi* de Matt Richardson et Shawn Wallace (éditions Eyrolles, 2014).

Et surtout n'oublie pas est que tu fais maintenant partie de la communauté mondiale Arduino. Tu trouveras plus de sources d'information et des tutoriels sur le site d'Arduino (`http://arduino.cc`) ainsi qu'un forum auquel participent de nombreux utilisateurs qui seront toujours prêts à répondre à tes questions.

Étape franchie : Tu es maintenant un expert en Arduino, bravo !

Copyright © 2015 Eyrolles.

Annexe A

Pour aller plus loin

CE N'EST PAS PARCE QUE tu es arrivé à la fin des aventures avec Arduino de ce livre que ton apprentissage s'arrête ici ! Tu peux maintenant continuer seul. Tu as encore plein de sujets à explorer et beaucoup de ressources peuvent t'y aider.

Plus de cartes, d'interfaces, de capteurs et d'actionneurs

Tu as déjà découvert deux cartes Arduino en plus de la Uno : la LilyPad et la Leonardo. Les cartes Arduino se présentent sous différentes formes et dimensions. Ainsi, si tu as un projet en tête, tu peux choisir la plus adaptée. Doit-elle être petite ? La SparkFun Arduino Pro Mini (`https://www.sparkfun.com/products/11113`) ou la Micro (`http://arduino.cc/en/Main/ArduinoBoardMicro`) devraient parfaitement convenir. Veux-tu utiliser beaucoup de capteurs et jouer des fichiers musicaux ? La Bare Conductive Touch Board (`www.bareconductive.com/shop/touch-board`) fait tout cela sur un seul circuit imprimé !

Copyright © 2015 Eyrolles.

Les cartes interfaces (shields)

Un « shield » est une carte à circuit imprimé prévue pour se placer exactement au-dessus d'une carte Arduino Uno. Elle peut comporter des circuits très complexes comme la carte Arduino Ethernet Shield (http://arduino.cc/en/Main/ArduinoEthernetShield), qui te permet de relier une carte Arduino Uno à Internet avec un câble Ethernet. Il existe également des shields permettant d'ajouter des écrans tactiles, des lecteurs MP3, des contrôleurs pour des moteurs et bien d'autres encore. Visite tes boutiques favorites, en magasin et en ligne, et regarde tout ce que tu peux te procurer.

Les capteurs et les actionneurs

Il y a également beaucoup plus de capteurs disponibles que ceux que tu as utilisés dans les aventures de ce livre. Il en existe pour à peu près tout ce que tu peux vouloir détecter ou mesurer avec ton Arduino. Tu veux mesurer la pression barométrique ? Teste le capteur BMP180 (www.adafruit.com/products/1603). Et en matière de sons ? Essaie le microphone Electret de chez SparkFun (https://www.sparkfun.com/products/9964).

Il en va de même pour les actionneurs. Il existe de nombreux types de moteurs et de haut-parleurs, et bien entendu, une large gamme de LED et d'écrans. La LED RVB Adafruit NeoPixel, par exemple, peut être pilotée avec un microcontrôleur Arduino (www.adafruit.com/category/168).

Il existe une alternative à l'achat de capteurs : réalise-les toi-même ! Le site Kobakant contient toute la documentation et les guides pour cela, souvent en utilisant des matériaux comme des cordelettes conductrices et du tissu (http://www.kobakant.at/DIY).

Tutoriels sur le Web

Le Web regorge de tutoriels et de sites d'informations. Seuls les plus populaires sont répertoriés ici, mais tu peux en trouver bien d'autres en cherchant !

Le site officiel d'Arduino

La première destination d'un ingénieur Arduino curieux est le site d'Arduino (http://arduino.cc). Tu y trouveras tout ce que tu as besoin de connaître au sujet des cartes Arduino officielles – et il en existe beaucoup. C'est aussi le site du terrain de jeu des passionnés d'Arduino (http://playground.arduino.cc) où les utilisateurs d'Arduino peuvent mettre à disposition leurs propres projets et tutoriels.

Si tu as des questions ou des problèmes pour réaliser un projet, tu peux les poser dans le forum Arduino (http://forum.arduino.cc). Tu trouveras toujours des utilisateurs Arduino prêts à t'aider. Mais attention, il est recommandé de prendre deux précautions au préalable. En premier, tu dois toujours vérifier que la question n'a pas

Copyright © 2015 Eyrolles.

déjà été posée en cherchant dans les forums, car quelqu'un a peut-être déjà apporté les informations dont tu as besoin. Ensuite, tu dois fournir le plus de détails possible sur ce que tu souhaites qu'il se passe et sur ce qui se passe réellement. Cela permet aux autres personnes de t'aider plus facilement et les incite davantage à de te répondre.

Le forum Arduino en français, que tu trouveras à l'adresse `http://forum.arduino.cc/index.php?board=33.0`, t'aidera si tu as des difficultés avec la langue anglaise.

Les sites de fabricants

En dehors du site web d'Arduino, les meilleures sources d'informations sont les deux fabricants de capteurs, d'actionneurs et de kits pour les microcontrôleurs Arduino : SparkFun et Adafruit. Les deux proposent d'excellents sites regorgeant de tutoriels et de guides. Adafruit est à l'adresse `https://learn.adafruit.com` et SparkFun à l'adresse `https://learn.sparkfun.com`. Ces deux sites sont en langue anglaise, mais en recherchant par exemple les mots clés « tutoriel Arduino », tu trouveras d'autres sites en français, également riches en informations. J'ai relevé les deux sites suivants auxquels tu peux te référer : `http://mchobby.be/wiki/index.php?title=Guide_d%C3%A9marrage_Arduino` et `https://tutoarduino.com/`. Bien d'autres existent, tu les trouveras sûrement avec un peu de patience.

Les blogs

Si tu es en manque d'inspiration pour ton prochain projet, essaie le blog Adafruit (`https://adafruit.com/blog`), le site du magazine *Make* (`http://makezine.com`) ou Hackaday (`http://hackaday.com`). Il existe également de nombreux blogs spécialisés. Par exemple, si tu t'intéresses aux projets réalisés sur des vêtements, rends-toi sur le site de Fashioning Technology (`http://fashioningtech.com`). Veux-tu faire voler ton Arduino ? DIY Drones (`http://diydrones.com/`) pourra t'informer sur ce sujet. Cette fois encore, je t'ai mentionné des blogs en langue anglaise, mais tu trouveras facilement des sites francophones, comme le site `http://arduino103.blogspot.fr/`.

Les vidéos

Parfois, rien ne vaut une démonstration vidéo des possibilités nouvelles. YouTube offre une multitude de clips sur les nouveaux capteurs et sur la façon de créer un projet novateur. Tu ne seras pas surpris d'apprendre qu'Adafruit (`https://www.youtube.com/user/Adafruit`) et SparkFun (`https://www.youtube.com/user/sparkfun`) ont leurs propres chaînes YouTube regorgeant d'informations. Le magazine *Make* en possède également une (`https://youtube.com/user/makemagazine`).

Dans YouTube, tu trouveras aussi une excellente série de vidéos en recherchant « Collin's Lab ». Collin Cunningham enseigne l'électronique de base avec une méthode facile à suivre. Certaines des vidéos sont réalisées en collaboration avec le magazine *Make* et d'autres avec Adafruit.

Copyright © 2015 Eyrolles.

Massimo Banzi, un des créateurs d'Arduino, produit par ailleurs une série de vidéos accompagnant leur kit de démarrage officiel. Ces vidéos appartiennent à un projet d'ensemble, mais tu peux te les procurer en achetant le kit. La première est à l'adresse `https://youtu.be/grU7eNPBRxk`.

Livres

Il est toujours utile d'avoir des livres papier à portée de main. Il en existe de nombreux et la plupart concernent les technologies les plus récentes. En voici quelques-uns qui t'aideront à débuter.

Sur Arduino

Il y a de nombreux projets que tu peux réaliser avec Arduino, en plus de ceux de ce livre. Lance-toi dans de nouveaux projets ! Construis davantage d'objets !

- *Getting Started with Arduino* par Massimo Banzi (Maker Media Inc., 2011)
- *Arduino For Dummies* par John Nussey (Wiley, 2013)
- *Arduino Projects For Dummies* par Brock Craft (Wiley, 2013)
- *Exploring Arduino: Tools and Techniques for Engineering Wizardry* par Jeremy Blum (Wiley, 2013)

Ces titres sont en anglais, mais tu trouveras également d'excellents ouvrages en français comme ceux-ci :

- *Le grand livre d'Arduino* par Erik Bartmann (Eyrolles, 2015)
- *Arduino pour les Nuls* par John Nussey (First éditions, 2015)
- *Maîtrisez les microcontrôleurs à l'aide d'Arduino* par Clemens Valens (Elektor, 2014)

Et bien d'autres...

Sur l'électronique de base

- *Practical Electronics for Inventors* par Paul Scherz et Simon Monk (Tab Books, 2013)

Et de nombreux livres en français, entre autres :

- *L'électronique en pratique* (deux volumes) par Charles Platt (Eyrolles, 2014-2015)
- *L'électronique pour les Nuls* par Gordon McComb (First éditions, 2014)
- *L'électronique pour les débutants qui sèchent les cours, mais soudent sans se brûler les doigts* par Rémy Mallard (Elektor, 2012)

Copyright © 2015 Eyrolles.

Sur les circuits souples et les projets vestimentaires

L'électronique et l'art de la création comme la couture vont bien ensemble. Si tu t'intéresses à cette association, voici quelques livres intéressants pour débuter :

- *Fashioning Technology* par Syuzi Pakhchyan (Maker Media, 2008)
- *Switch Craft: Battery-Powered Crafts to Make and Sew* par Alison Lewis et Fang-Yu Lin (Potter Craft, 2008)
- *Sew Electric* par Leah Buechley, Kanjun Qi et Sonja de Boer (HLT Press, 2013)
- *Make: Wearable Electronics* par Kate Hartman (Maker Media, 2014)

Autres sujets spécialisés

Il y a bien entendu beaucoup d'autres choses qui peuvent être fabriquées avec un Arduino. Pourquoi pas un Arduino pilotant des robots ou communiquant sur Internet ?

- *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists* par Dustyn Roberts (McGraw-Hill, 2011)
- *Making Things Talk: Using Sensors, Networks, and Arduino to See, Hear, and Feel Your World* par Tom Igoe (Maker Media, 2011)

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Annexe B

Où se procurer outils et composants

RECHERCHER SEUL les outils et les composants électroniques nécessaires peut être difficile. Par chance, grâce au nombre grandissant des systèmes électroniques à réaliser soi-même, tu n'auras aucun mal à acheter ce dont tu as besoin pour tes projets Arduino.

Les kits de démarrage

Acheter un kit de démarrage est une manière vraiment facile de commencer les aventures de ce livre. Ce kit comprend une carte Arduino Uno et à peu près tous les composants dont tu as besoin tels que les LED, les résistances et un servomoteur. Tu peux comparer la liste des éléments nécessaires dans l'introduction de ce livre avec ce qui est inclus dans le kit de démarrage, et acheter en plus ce qui n'y figure pas. Tu devras de toute façon acheter séparément une Arduino Leonardo et une Arduino LilyPad USB pour réaliser les aventures 7 et 8.

La plupart des magasins cités dans cette annexe vendent leurs propres kits, les alternatives sont donc nombreuses. La société Arduino commercialise aussi ses kits, accompagnés d'un livre (en anglais) consacré à des projets. Tu peux l'acheter directement sur la boutique Arduino en ligne (`http://store.arduino.cc/product/K000007`).

Copyright © 2015 Eyrolles.

Les magasins spécialisés

Dans les boutiques spécialisées, tu peux choisir les éléments dont tu as besoin, ce qui n'est pas sans intérêt. Par exemple, quand tu utilises des composants physiques, il est souvent utile que tu puisses les voir par toi-même. De plus, tu peux poser des questions aux vendeurs, qui sont toujours prêts à t'aider. Et surtout, tu n'as pas à attendre que ton colis te parvienne, tu peux rentrer chez toi et commencer à t'en servir immédiatement !

En France, dans ses magasins, Selectronic (`www.selectronic.fr`) vend toutes sortes de composants qui sont en stock. Tu peux maintenant y acheter des cartes Arduino tout comme les composants électroniques, ainsi que la plupart des composants miniatures tels que les LED et les résistances. Tu peux consulter leur catalogue papier ou leur boutique en ligne pour sélectionner les références des composants dont tu as besoin, un vendeur te les délivrera. Ils stockent également les outils indispensables comme les fers à souder.

Les boutiques en ligne

Il existe deux sortes de boutiques en ligne où trouver des composants électroniques : les sites spécialisés pour les amateurs et les sites destinés aux professionnels ayant un catalogue très vaste. Si tu débutes, il est préférable que tu t'adresses à une boutique destinée aux amateurs qui pourra te fournir tout ce dont tu as besoin.

Les plus grandes boutiques en ligne ont des catalogues recélant des milliers, voire des millions de références de composants et il peut être difficile de s'y retrouver si tu ne sais pas exactement ce que tu recherches. Ils pratiquent en revanche des prix généralement inférieurs à ceux des autres boutiques et stockent des composants moins populaires qui sont parfois difficiles à trouver.

Les boutiques en ligne expédiant depuis l'Europe

Adafruit et SparkFun sont les meilleurs sites de référence. Ils sont basés aux États-Unis, mais heureusement, il existe de nombreux distributeurs européens qui importent leurs produits. Cela te permet d'obtenir plus rapidement ce que tu commandes ! Tu peux essayer les sites suivants pour vérifier s'ils ont ce dont tu as besoin :

- Selectronic : `www.selectronic.fr`
- Conrad électronique : `www.conrad.fr`
- Arduino store : `http://store.arduino.cc`
- Cool Components : `www.coolcomponents.co.uk`
- Maplin : `www.maplin.co.uk`
- Oomlout : `http://oomlout.co.uk`

Copyright © 2015 Eyrolles.

- Pimoroni : http://shop.pimoroni.com
- Proto-Pic : http://proto-pic.co.uk
- RobotShop : http://www.robotshop.com/eu/fr
- SK Pang : http://skpang.co.uk

Les boutiques ayant des catalogues professionnels qui expédient en Europe :

- Digi-Key : www.digikey.fr
- Farnell : http://fr.farnell.com
- Mouser : www.mouser.fr
- Rapid : www.rapidonline.com
- RS Components : http://fr.rs-online.com/web

Les boutiques en ligne expédiant depuis les USA ou le Canada

Les deux sites les plus spécialisés pour les makers sont Adafruit (www.adafruit.com) et SparkFun (https://www.sparkfun.com). Ils ont tous deux d'excellents guides et tutoriels (en anglais) pour te permettre d'utiliser ce qu'ils vendent. Leurs offres se recoupent partiellement, mais chacun fabrique ses propres produits. Ils sont situés aux États-Unis, mais si tu ne souhaites pas attendre longtemps tes commandes et payer des frais d'expédition élevés, réfère-toi à la section précédente qui indique les boutiques expédiant depuis la France ou l'Europe.

Deux autres sites incontournables

Les amateurs d'électronique et de robotique, dont tu fais maintenant partie, font rarement leurs achats sans consulter au préalable ces deux sites de vente en ligne qui leur sont devenus incontournables, parfois irremplaçables. La diversité des matériels, outils et kits proposés est importante, et les prix sont souvent très compétitifs. Ils offrent toutes sortes de produits neufs ou d'occasion dans tous les domaines, y compris bien entendu tout ce qui concerne l'électronique, les microcontrôleurs et la robotique. Les expéditions ne sont pas toujours rapides, mais elles sont généralement bon marché, car les produits sont en provenance de Chine le plus souvent. Tu devras parfois prévoir de payer des frais de douane, en particulier pour des achats d'un montant dépassant quelques euros.

Le premier site à consulter sans modération est bien entendu eBay (www.ebay.fr), dans lequel tu trouveras des fournisseurs français, de l'Union européenne, américains et asiatiques. Seuls ces derniers peuvent nécessiter le paiement de frais d'importation, mais ils disposent souvent d'entrepôts en Angleterre, en Allemagne et parfois en

Copyright © 2015 Eyrolles.

France, évitant l'application de frais de douane et des expéditions trop longues.

Également incontournable par la richesse de ses offres, le site Aliexpress (www.aliexpress.com) est une plate-forme d'achat qui regroupe des vendeurs chinois et gère leurs envois ainsi que le règlement unique de leurs ventes. Les prix sont le plus souvent indiqués envoi compris, ils sont exprimés en euros et le site est disponible en langue française. Le seul inconvénient est que les expéditions peuvent prendre de 10 à 30 jours. Il faut donc être patient, juste le temps de consulter les informations techniques des produits achetés pour être prêt le jour de la réception, car ces sites ne délivrent que très rarement la documentation de ce qu'ils commercialisent.

Attention, des taxes douanières peuvent s'appliquer à ces achats et il vaut mieux te faire aider d'un adulte pour les effectuer, car ils sont toujours payables par carte de crédit ou par Paypal. Je te conseille d'ailleurs d'utiliser des numéros d'e-carte bleue, c'est-à-dire des numéros uniques pour un achat unique d'un montant limité et à durée également limitée (environ 20 minutes). Ce moyen de paiement, qui évite tout piratage des numéros de carte de crédit, est en général offert par les banques françaises.

Copyright © 2015 Eyrolles.

Glossaire

Actionneur Composant transformant un signal électrique en une action dans son environnement comme de la lumière, du son ou un mouvement.

Analogique (signal) Signal variant entre les niveaux `HIGH` (HAUT) et `LOW` (BAS), à l'opposé des signaux numériques. Sur la carte Arduino Uno, un signal analogique s'évalue en entrée par un nombre variant entre 0, pour le niveau bas de la masse GND (0 V), et 1023, pour une tension de 5 V. Et en sortie, un signal analogique peut être délivré à une valeur variant entre 0 pour le 0 V et 255 pour le 5 V.

Anode Fil positif d'un composant polarisé ou directionnel, comme le fil le plus long d'une LED.

Argument Élément d'information fourni à une fonction que celle-ci utilise pour effectuer les actions de son programme. L'argument se place entre les parenthèses suivant le nom de la fonction. Par exemple, la fonction `delay(1000)` a pour argument `1000`, qui représente le nombre de millisecondes que tu souhaites comme durée de pause de ton microcontrôleur Arduino, avant qu'il n'exécute la ligne de code suivante.

Bibliothèque C'est un ensemble de fonctions réutilisables qui peuvent être importées et utilisées dans tes sketches.

Binaire (nombre) Nombre qui n'utilise que les chiffres 0 et 1, contrairement à un nombre décimal qui utilise les chiffres de 0 à 9. Un nombre binaire est aussi appelé nombre en base 2, un nombre décimal étant appelé nombre en base 10.

Bouton-poussoir à montage en façade C'est un bouton-poussoir prévu pour être placé à l'intérieur du boîtier d'un appareil. Il est muni d'un écrou et d'une rondelle permettant de le fixer au travers d'un trou percé dans la façade de l'appareil.

Bouton-poussoir tactile Type particulier d'interrupteur. Un bouton-poussoir normalement fermé interrompt le courant dans un circuit lorsqu'il est appuyé. Un bouton-poussoir normalement ouvert fait l'inverse : il interrompt le courant uniquement lorsqu'il n'est pas appuyé.

Câble à pinces crocodiles Fil électrique muni à ses extrémités de pinces à ressorts qui ressemblent à des mâchoires de crocodile. Ils sont utiles pour réaliser des

Copyright © 2015 Eyrolles.

prototypes de circuits souples ou pour relier des composants qui n'acceptent pas les fils de pontage.

Capacitance Propriété qui permet de stocker une charge électrique. Les composants électriques spécialement conçus pour exploiter cette propriété sont les condensateurs. Mais d'autres objets ont une capacitance, et même les êtres humains !

Capteur Composant sensible à l'environnement qui l'entoure tel que la lumière, le son ou un mouvement, qu'il traduit en un signal électrique. On peut citer comme exemples les potentiomètres et les photorésistances.

Cathode Pôle négatif d'un composant polarisé ou directionnel, par exemple, le fil le plus court d'une LED.

Circuit intégré (*Integrated Circuit* ou IC) Ensemble de circuits électroniques complexes incorporés dans un seul boîtier. Un même circuit intégré peut se présenter sous différentes formes appelées boîtiers. Lorsque tu réalises ton circuit sur une plaque de montage, tu dois utiliser les circuits intégrés de type DIP ou DIL. C'est une forme de boîtier qui permet d'insérer les broches de connexion dans les trous de ta plaque de montage.

Circuit souple Circuit réalisé avec des matériaux souples, comme de la cordelette conductrice et du tissu. Les circuits souples sont en général utilisés dans des projets destinés à de l'habillement.

Commentaire Annotation dans ton code qui explique ce qu'une ligne ou une portion de programme est supposée faire. Chaque commentaire doit commencer par `//` ou, si tu veux écrire un commentaire sur plusieurs lignes, entre `/*` et `*/`. Ces caractères spéciaux indiquent à l'ordinateur qui exécute le programme d'ignorer la ou les lignes considérées.

Communication série Technique permettant l'échange de données en réception et émission entre deux appareils, comme une carte Arduino et un ordinateur. Un seul élément des données est transmis à la fois, de façon séquentielle.

Compilation Processus qui consiste à prendre le code écrit par un être humain et à le convertir en instructions compréhensibles par un ordinateur.

Composant monté en surface, CMS (*Surface Mount Device* ou SMD) Format possible pour le boîtier d'un circuit intégré ou de composants comme les résistances. Il est prévu pour être soudé directement sur une surface plane sans nécessiter d'insérer des fils ou des broches dans les trous d'un circuit imprimé.

Courant Exprime l'intensité de l'énergie électrique passant dans un conducteur. C'est l'équivalent électrique de l'écoulement hydraulique dans un tuyau. Le courant se mesure en ampères (A). Les courants les plus faibles se mesurent en milliampères (mA).

Copyright © 2015 Eyrolles.

Courant continu (*Direct current* ou DC) Type d'énergie électrique utilisée dans les circuits Arduino. C'est le même type de tension que celle fournie par une pile électrique et c'est le contraire d'une tension alternative (*Alternating Current* ou *AC*), qui est présente sur les prises électriques de ton logement.

Débogage Processus de recherche des causes des erreurs dans les programmes et de leur correction.

Déclaration Action consistant à créer une nouvelle variable en lui donnant un nom et un type de donnée, par exemple `int` pour des nombres entiers. La variable ne contient aucune valeur avant qu'une première valeur ne lui soit affectée.

Diviseur de tension (pont) Circuit qui délivre en sortie une fraction de la tension qui lui est appliquée en entrée. C'est un circuit très utile pour traduire le changement de valeur d'une résistance en variation de tension.

Dual in-line package (DIP ou DIL) Type de boîtier de circuit intégré. Il possède deux rangées de broches qui peuvent s'insérer dans une plaque de montage.

Entrée flottante Broche d'entrée qui n'est reliée à rien. La broche lit des valeurs aléatoires si elle n'est pas reliée à une source de tension comme la masse GND, 5 V ou un capteur.

float (type de donnée) Type de donnée des nombres qui ne sont pas des nombres entiers mais possèdent une virgule et des décimales, par exemple 1,3 et -54,089.

Fonction Ensemble de lignes de code regroupées sous un même nom. Une fonction peut être utilisée de façon répétitive. Elle peut recevoir des informations en entrée (arguments) et fournir d'autres informations en sortie quand elle se termine. Néanmoins, toutes les fonctions ne font pas cela.

for (boucle) Élément de programmation qui permet de répéter un bloc d'instructions un nombre prédéterminé de fois.

IDE (*Integrated Development Environment*) Logiciel d'application qui est utilisé pour écrire le code de programmation dans un langage particulier. On le nomme parfois Environnement de Programmation. Cette application permet de créer et de modifier du code et de l'exécuter. De nombreux IDE disposent de fonctions d'aide permettant aux programmeurs de déboguer leurs programmes – en d'autres termes, de rechercher les erreurs dans leurs programmes.

Instanciation Action qui consiste à fournir une valeur à une variable pour la première fois. L'instanciation peut être faite au moment de la déclaration de la variable ou par la suite, mais la déclaration doit toujours être effectuée en premier. On utilise aussi le nom initialisation pour qualifier cette action.

Interrupteur Composant qui interrompt ou redirige le courant d'un circuit.

Copyright © 2015 Eyrolles.

LED (*Light Emitting Diode*) Composant électronique qui fournit de la lumière lorsqu'un courant électrique la traverse. Une diode ne laisse passer le courant que dans un seul sens. Les diodes LED ne s'illuminent que lorsque leur fil le plus long est relié au côté positif d'une source d'alimentation électrique et leur fil le plus court au côté négatif de cette source. Si les connexions des fils sont inversées, la LED ne s'allume pas.

LED RVB (rouge-vert-bleu) LED dans un boîtier comportant quatre fils et contenant trois lampes : une rouge, une verte et une bleue. Les trois lampes partagent une anode ou une cathode commune.

Loi d'Ohm Relation mathématique entre la tension, l'intensité du courant et la résistance d'un circuit. La tension est égale au courant multiplié par la résistance – ou, sous forme abrégée, V=IR.

long (type de donnée) Type de donnée permettant de stocker un nombre entier compris entre -2 147 483 648 et 2 147 483 647.

Modulation en largeur d'impulsions, MLI (*Pulse Width Modulation* ou *PWM*) Technique utilisée par un microcontrôleur Arduino pour générer un signal compris entre 0 et 5 V. Le signal est une impulsion passant rapidement du niveau `LOW` au niveau `HIGH` permettant à la tension de sortie résultante de se situer entre ces deux tensions.

Numérique (signal) Signal ne pouvant prendre pour valeur que 0 ou 1 ou `HIGH` ou `LOW`. Sur une Arduino Uno, un signal `HIGH` vaut 5 V et un signal `LOW` est au niveau de la masse GND (0 V).

Photorésistance Résistance dont la valeur change selon la luminosité qu'elle reçoit.

Piézoélectrique (composant) Cristal qui se dilate ou se contracte sous l'influence d'un courant électrique le traversant. Il peut aussi générer une tension électrique lorsqu'il est compressé puis relâché, ou déformé.

Pilote (driver) Petit logiciel qui permet à ton ordinateur de communiquer avec un appareil externe, comme une imprimante ou un clavier.

Plaque de montage Composant réutilisable qui te permet de créer des circuits sans devoir souder les composants. Les plaques de montage possèdent des rangées de trous dans lesquels tu insères les composants formant ton circuit.

Potentiomètre Type de résistance variable comportant un axe rotatif qui permet de faire varier la valeur de sa résistance.

Rapport cyclique Rapport entre la durée d'un signal au niveau `HIGH` et sa durée au niveau `LOW` dans un cycle donné. En modulation en largeur d'impulsions (*Pulse*

Copyright © 2015 Eyrolles.

Width Modulation, *PWM*, ou Modulation en Largeur d'Impulsions, MLI), plus le rapport cyclique est élevé, plus la tension de sortie est élevée.

Registre à décalage Composant permettant le contrôle de sorties multiples par un nombre d'entrées restreint. Il est utilisé de façon fréquente pour piloter un grand nombre de LED.

Résistance Composant électrique qui s'oppose au passage d'un courant dans un circuit. Une LED pourrait, par exemple, être détériorée par un courant trop fort, si tu ajoutes une résistance de valeur appropriée au circuit de la LED pour limiter l'intensité du courant, la LED est protégée. La résistance se mesure en ohms, ou Ω. Tu dois choisir une résistance ayant une valeur correcte pour limiter le courant traversant un circuit. La valeur d'une résistance est indiquée par des bandes de couleurs qui s'interprètent de gauche à droite.

Résistance de maintien au niveau haut Résistance reliée à la tension d'alimentation d'un circuit qui permet de fixer le niveau par défaut d'une broche de ce circuit au niveau `HIGH`. La valeur de cette résistance est habituellement de 10 kΩ.

Saut de ligne (caractère de) Caractère représentant ce qui se passe quand tu appuies sur la touche Entrée ou Retour de ton clavier.

Servomoteur Moteur dont on peut contrôler la rotation à une position particulière. Sa rotation est en général limitée à un angle de 180 degrés.

Sketch Programme Arduino. Appelé aussi parfois croquis.

Tableau Liste d'éléments de même type dans un programme. Un tableau peut, par exemple, mémoriser une liste de nombres entiers de type `int`.

Tableau à deux dimensions Données organisées en rangées et colonnes, à la manière de la feuille de calcul d'un tableur.

Tension (ou voltage) Elle exprime la différence de potentiel électrique entre deux points d'un circuit. Elle est équivalente à la pression de l'eau dans des tuyaux, c'est cette pression qui provoque le passage du courant dans un circuit. La tension est exprimée en volts (V).

Variable Élément de codification d'un programme qui mémorise une valeur pouvant être modifiée. Par exemple, la variable `LEDVerte` contient le nombre 5.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Index

Symboles

{} 40
\; 54
&& 78
< 69
< > 212
> 78
74HC595 122
#define 174
#include 91, 211, 278

A

actionneur 88, 262
  définition 88, 263
adaptateur
  fiche d'alimentation cylindrique 10
alimentation électrique 10
ampère 42
analogRead() 68, 192
analogWrite() 192, 194
anode 199
Arduino 1
  bouton Reset 65
  broches numériques 47
  cartes 2
  IDE 23
  installer le logiciel 16
  installer l'environnement de programmation 2
  kit de démarrage 327
  LED intégrée 37
  Leonardo 2, 17, 227
  LilyPad 258
  LilyPad Simple 258
  LilyPad Simple Snap 258
  LilyPad USB 2, 17, 258
  logiciel 15
  télécharger logiciel 23
  téléverser 15
  type de carte 31
  Uno 2, 22
arduino.inf 26
Arduino Leonardo 227
  arrêter son fonctionnement 232
  comportement clavier 230
  particularités 230
Arduino LilyPad SimpleSnap 282
Arduino LilyPad USB 257
  batterie 276
  comportement 258
  interrupteur 277
  interrupteur de mise en marche 261
  recharge batterie 260, 277
Arduino Uno
  broches analogiques 66
Atmel 34
avrdude 34, 56

B

barrette de connexion 7
batterie
  capacité 276
  LiPo 10, 276
bibliothèque de code 17, 191, 208
  définition 209
bit 143
BlinkFonctions.ino 124
blogs sur Arduino 323
Bonjour tout le monde 30, 260
boolean 302, 306
boucle for 17
boutiques en ligne 328
bouton
  Enregistrer 54
  Moniteur série 63
  Téléverser 56
bouton-poussoir 6
  pour montage en façade 101
  tactile 6, 88, 95
    définition 96
broche
  analogique 194
  entrée analogique 68
broche +5 V 44
broche GND 43

C

câble
  à pinces crocodiles 263
  micro USB 2, 228
  standard 2
  USB 2
capacité électrique 208
capacitiveSensor() 212
CapacitiveSensor 210
capteur 1, 88
  accéléromètre 235
  capacitif 208
  définition 88, 262
  de lumière 235
  de pression barométrique 322
  microphone 235
  piézoélectrique 285
  sensitif 191
caractère retour à la ligne 61
carte FTDI 259
cathode
  commune 200
  définition 199
circuit intégré 122
  boîtier DIP 122
  boîtier N 122
  définition 130
circuit souple 257
  définition 263
ciseaux 14
commentaire 19, 126
  définition 36
compiler 29
composant monté en surface 131
connecteur USB 229

Copyright © 2015 Eyrolles.

const 266
conversion binaire-décimal 142
convertisseur binaire décimal 155
cordelette conductrice 11, 17
courant continu 89
courant électrique 42
couteau 12
cutter 12

D

déboguer 60
déclaration de variable 162
delay() 38, 39
détection des vibrations 287
digitalRead() 192, 194
digitalWrite() 39, 169, 192, 194
DIL 130
DIP 130
dossier libraries 209
drapeau 303

E

écran tactile 322
écrou 118
élément piézoélectrique 169
else 303
  bloc d'instructions 304
entrée analogique
  fonctionnement 235
entrée flottante 97
esperluettes 78
Ethernet
  carte Arduino 322

F

fer à souder 11
fichier en-tête (.h) 277
fil
  à souder 9
  de pontage 4
  émaillé 9
  isolant 11
  multibrins 8
  muni de pinces crocodiles 10
  rigide 8
fil à coudre conducteur 280
float
  type de donnée 162
fonction 17, 123
  anatomie 125
  argument 127
  définition 37
  parenthèses 125
  utilisation 123
for 164
  anatomie d'une boucle for 129
  combinée avec tableau 164
  utilisation 128
Fritzing 15
FTDI 259

G

guillemets 62

H

hertz 168
HIGH 39

I

IDE 22
  bouton 28
  menus français 28
  visite 28
if 245, 303
  bloc d'instructions 245
  instruction 69
initialiser
  définition 162
INPUT 232
INPUT_PULLUP 98
int 57
  type de donnée 124
interrupteur
  définition 95

K

Keyboard.begin() 233
Keyboard.press() 250
  KEY_DOWN_ARROW 250
  KEY_LEFT_ARROW 250
  KEY_RIGHT_ARROW 250
  KEY_UP_ARROW 250
Keyboard.println() 233
Keyboard.releaseAll() 250
kit de démarrage 327

L

langage C/C++ 2
LED 4
  contrôle de brillance 195
  définition 30
  fil court 46
  fil long 45
  LilyPad 4, 264
  RGB 199
  RVB 199
  sur carte Arduino 30
  tricolore 17
loi d'Ohm 236
long
  type de donnée 212, 302
loop() 37
LOW 39

M

manette de jeu 240
map() 221
masse 43
mélange
  additif 207
  de couleurs 207
  soustractif 207
millis() 302
modulation de largeur d'impulsion (MLI) 197
  exemples 198
  rapport cyclique 198
moniteur série 63
moteur 89
  différents types 89
  pas à pas 89
multimètre 13, 179, 185
  calibre automatique 13
  test de conductivité 185
  test de continuité 13

N

nombre binaire 142
nombre décimal 142
noTone() 170

O

octet 143
ohm 5, 42
ordinateur
  Linux 16, 23
  Mac 16, 23
  Windows 16, 23

P

papier mâché 222
persistance rétinienne 273
photorésistance 228, 235
piézoélectrique 169
pile 10
pilote 23, 230
  mise à jour 26
pince
  à dénuder 12
  coupante 12
  pliante 12
pinMode() 38, 39
  OUTPUT 38
pitches.h 174
plaque de montage 3, 11, 44
  lignes d'alimentation 45
pont diviseur de tension 237

Copyright © 2015 Eyrolles.

port
Mac 33
Ports (COM & LPT) 26
potentiomètre 5, 17, 65
ajustable 65
lecture valeur 67
programme 123
prototype 72, 102

R

rapport cyclique 198
Raspberry Pi 2, 27
réétalonnage 221
registre à décalage 6, 122, 131
brochage 134
CLOCK 132
DATA 132
fonctionnement 132, 133
LATCH 132
reproducteur piézoélectrique 170
reproduction sonore 169
résistance 5, 97
de limitation de courant 43
de maintien 97
électrique 42
rondelle 118

S

schéma
interconnexions 102
Serial.begin() 60
Serial.print() 61
Serial.println() 61
Servo
attach() 92
type de donnée 91
write() 93
Servo.h 90
servomoteur 6, 90
setup() 37
shield 322
carte Arduino Ethernet 322
shiftOut() 140
LSBFIRST 140
signal analogique 191, 193
signal numérique 191, 193
OFF 193
ON 193
sketch 16
09.USB 255
AnalogRead 238
AnalogReadSerial 68
Blink 124
clignotement (blink) 31
DigitalReadSerial 96
Fading 195
Knock 288
nouveau 54
Sweep 90
tone-Melody 174
son
diffusion 159
production 168
reproduction 169
soudure 9, 11, 80
système d'exploitation 23

T

tableau 159, 161, 257
à deux dimensions 265
fonctionnement 257
lignes et colonnes 265
liste 265
multidimensionnel 265
premier élément 161
téléverser 29
tension électrique 42
terre 43
tone() 170, 183, 292
trame 267, 270
transducteur piézoélectrique 7, 17
transmission série 60
TRUE 78

V

variable 54
définition 57
règles des noms 57
utilisation 160
vibration
détecter 285
vidéo 323
void
type de donnée 124
volt 42

W

while() 231